

HSI 2020: Advanced Computational Chemistry

# Applied Machine Learning for Chemistry

Aug 26: 10:30~12:00 (90min)

Aug 26: 13:00~14:30 (90min)

Ichigaku Takigawa

[ichigaku.takigawa@riken.jp](mailto:ichigaku.takigawa@riken.jp)

<https://itakigawa.github.io>



You can get this slide PDFs and more.

The image shows a Google search for 'Ichigaku Takigawa'. The search results show 'itakigawa.github.io' as the top result. A red dashed oval highlights the URL 'https://itakigawa.github.io/hsi2020/' in the browser's address bar. A black arrow points from the search result to the browser window. The website content includes the title '少年易老學難成' (Shounen Eiro Gakushu Nanijiru) and the Latin motto 'Ars longa, vita brevis, occasio praeceps, experimentum periculosum, iudicium difficile.' Below the title is a photo of a person and the name 'たきがわ いちがわ 一学' (Takigawa Ichigawa Gaku).

<https://itakigawa.github.io/hsi2020/>

# Brief Bio: Ichigaku TAKIGAWA



Computer Scientist (Machine Learning)

10 years@**Hokkaido U**  
(1995~2004)

Statistical Machine Learning & Signal Processing  
(**Dept. Engineering**)

7 years@**Kyoto U**  
(2005~2011)

**Bioinformatics** (**Institute for Chemical Research**)

**Chemoinformatics** (**Dept. Pharmaceutical Sciences**)

7 years @**Hokkaido U**  
(2012~2018)

Machine Learning (**Dept. Information Science and Technology**)

JST Presto (**Advanced Materials Informatics Group**)

? years@**RIKEN**  
(2019~)

Medical-risk Avoidance based on iPS Cells Team (**RIKEN Center for Advanced Intelligence Project**)

Institute for Chemical Reaction Design and Discovery  
(**Hokkaido U**)

## Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

## Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
  - "The quality of your inputs decide the quality of your output"
  - Training / validation / test data
  - Tuning hyperparameters
  - Identification and design of input variables (or "descriptors")
  - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
  - "The quality of your inputs decide the quality of your output"
  - Training / validation / test data
  - Tuning hyperparameters
  - Identification and design of input variables (or "descriptors")
  - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

# "machine learning" is a new way of programming

Consider when you need to write a code for a "Rock paper scissors" robot.



# "machine learning" is a new way of programming

Your first task would be to write a code for computers to recognize the hand shapes among rock, paper, or scissors.



# "machine learning" is a new way of programming

But you'll instantly recognize this task is really really hard... we need to consider many variations and nuisances... but human can do this easily.





# "machine learning"

**Concerned with the question of how to construct computer programs that automatically improve with experience.**

———— Tom Mitchell

Tasks below would need experience rather than a single principle.

- Learning to recognize spoken words, handwritten characters, etc
- Learning to recognize who is who by seeing faces
- Learning to walk, speak, swim, ski, etc.
- Learning to drive an autonomous vehicle
- Learning to play world-class go, chess, shogi, etc.

# "machine learning"

“Machine” means **computer programs**

“Learning” means to **automatically improve with experience**

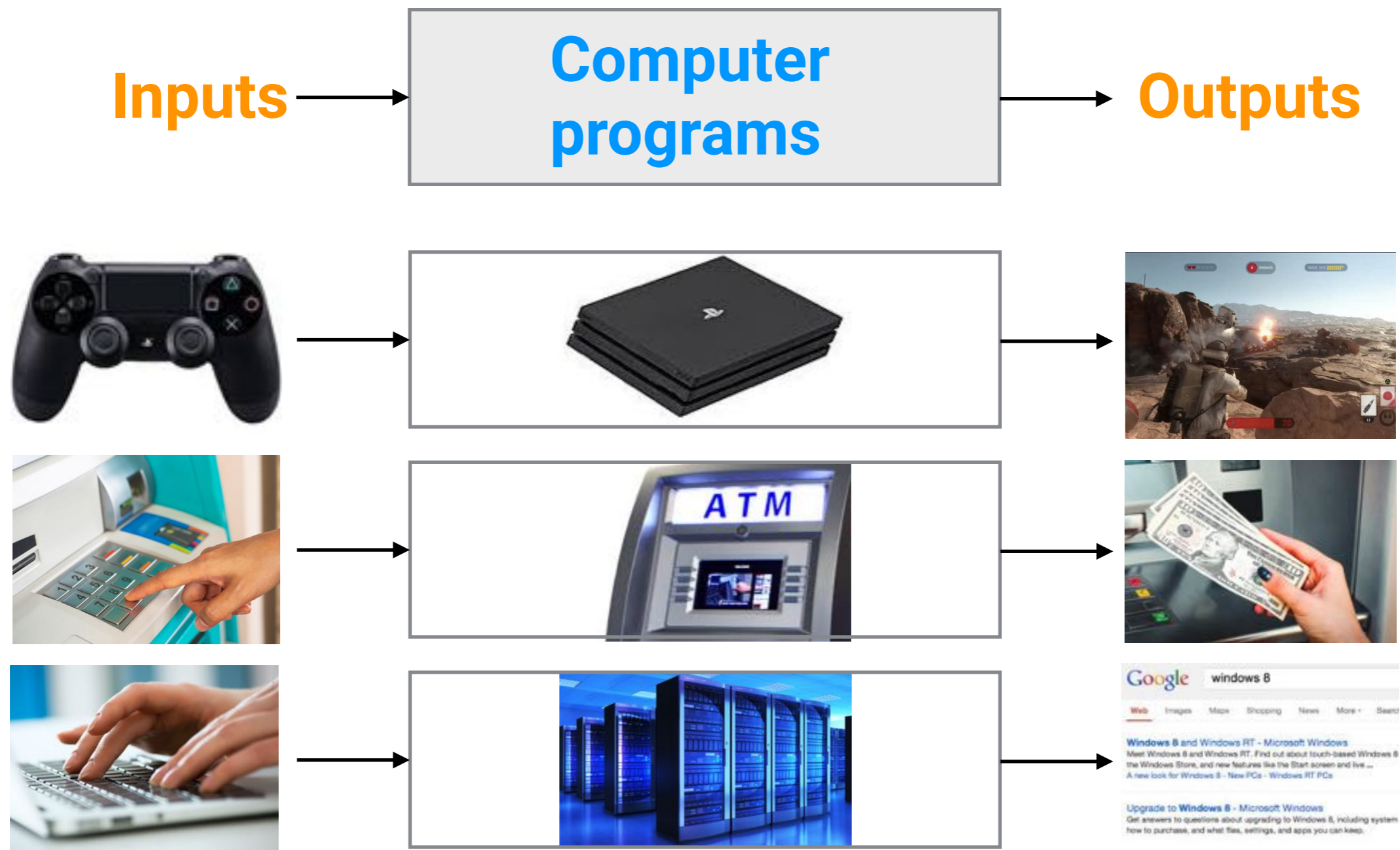
By machine learning, let's get **computer programs** to **automatically improve with experience**

But what exactly means....

- **Computer programs?** 🤔
- **Automatically improve with experience?** 🤔

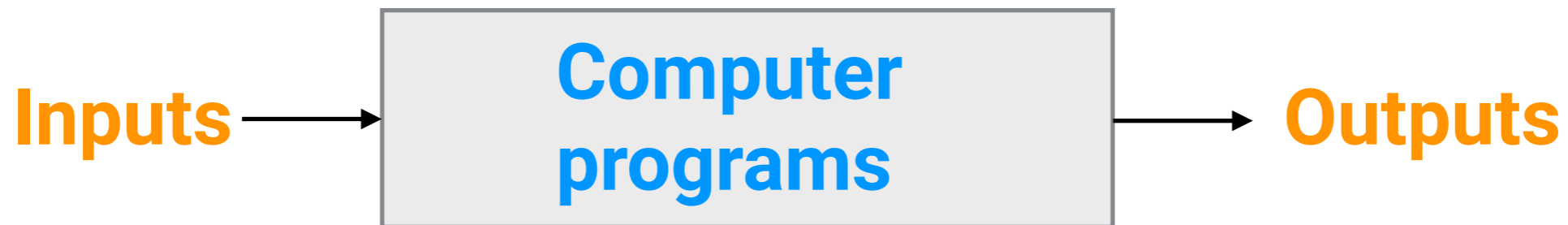
# Computer programs

Any **computer programs** process **inputs** to get **outputs**.



# Computer programs

Any **computer programs** process **inputs** to get **outputs**.



- We need to **explicitly know the complete procedure** to get outputs from inputs. 😱
- We **manually code the computer program** using computer programming languages. 🤯

But often we don't know how to do it 😞

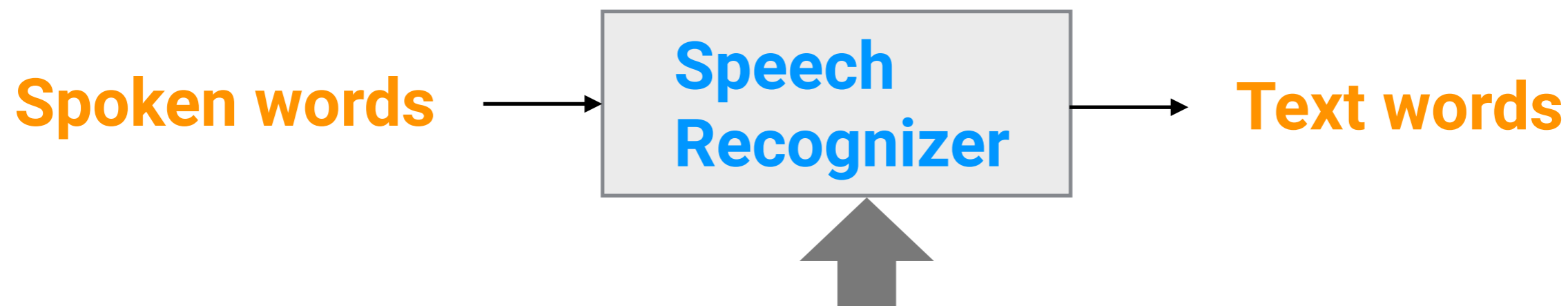


Consider how we can construct **computer programs** that can

- Learn to recognize spoken words, handwritten characters, etc
- Learn to recognize who is who
- Learn to walk, speak, swim, ski, etc.
- Learn to drive an autonomous vehicle
- Learn to play world-class go, chess, shogi, etc.

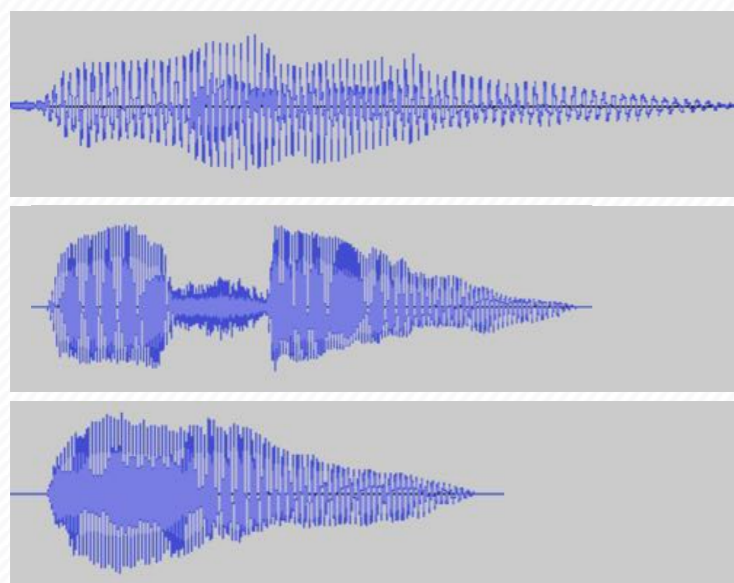
# So we give “training data” to teach programs

“automatically improve with experience (given data)”



“training data” = examples of input-output pairs

**Inputs**



**Outputs**

“hello”

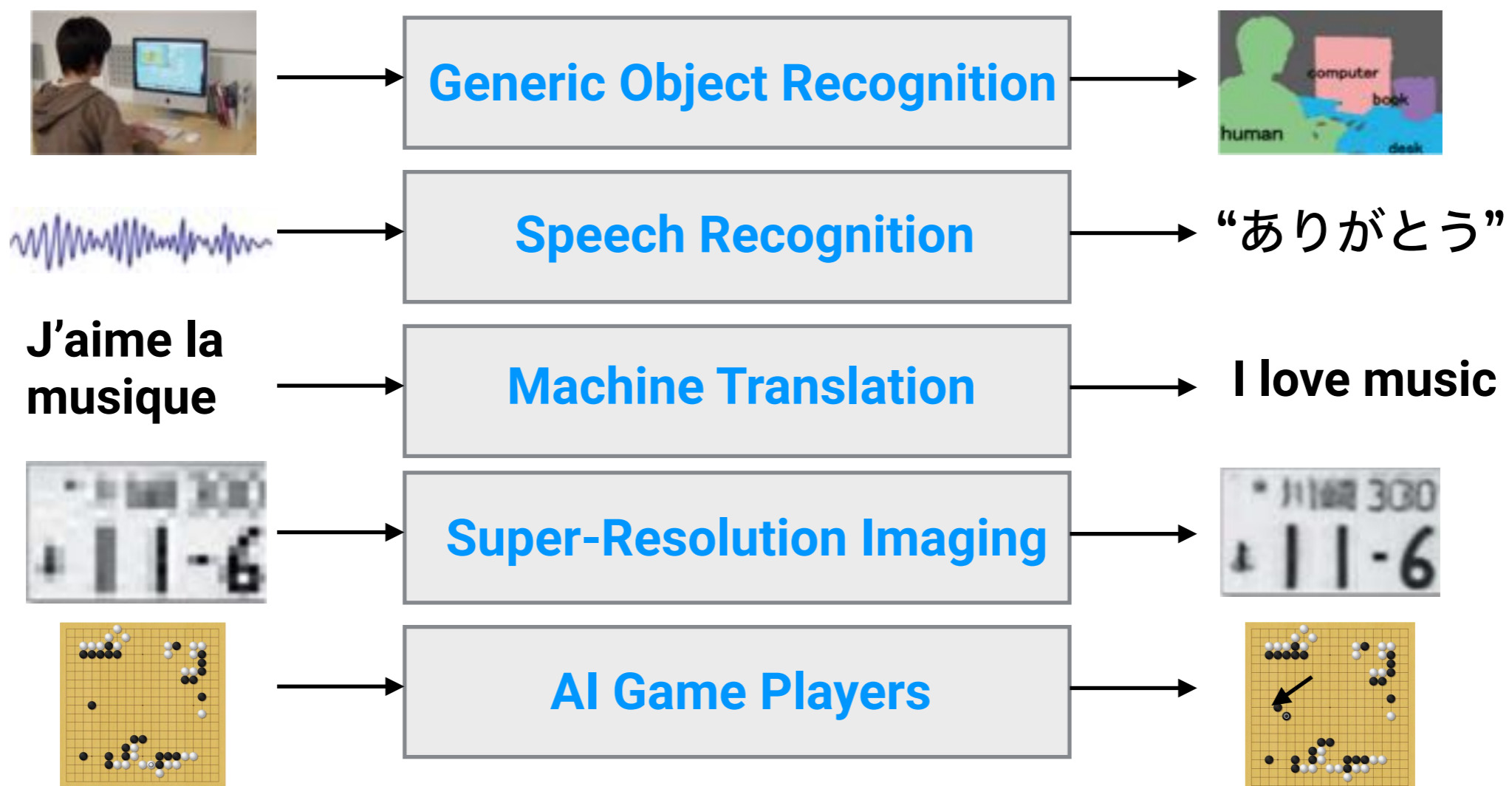
“machine”

“learning”

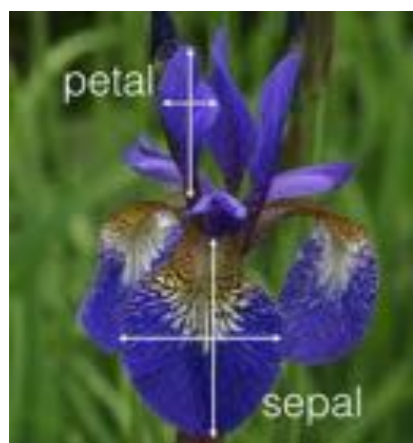
⋮

# (Supervised) machine learning

Machine learning is a way to construct **a computer program** directly by a given (large) collection of **input-output** examples **without being explicitly programmed**.



# More typical cases with tabular data



Inputs

$x$

ML model

Outputs

$y$

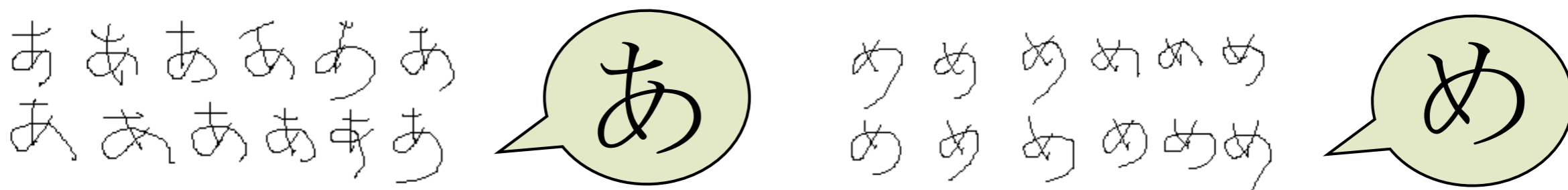


	A	B	C	D	E	F
1	ID	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
2	8	5	3.4	1.5	0.2	setosa
3	51	7	3.2	4.7	1.4	versicolor
4	36	5	3.2	1.2	0.2	setosa
5	15	5.8	4	1.2	0.2	setosa
6	60	5.2	2.7	3.9	1.4	versicolor
7	88	6.3	2.3	4.4	1.3	versicolor
8	126	7.2	3.2	6	1.8	virginica
9	32	5.4	3.4	1.5	0.4	setosa
10	13	4.8	3	1.4	0.1	setosa
11	146	6.7	3	5.2	2.3	virginica
12	5	5	3.6	1.4	0.2	setosa
13	105	6.5	3	5.8	2.2	virginica
14	133	6.4	2.8	5.6	2.2	virginica
15	92	6.1	3	4.6	1.4	versicolor
16	59	6.6	2.9	4.6	1.3	versicolor



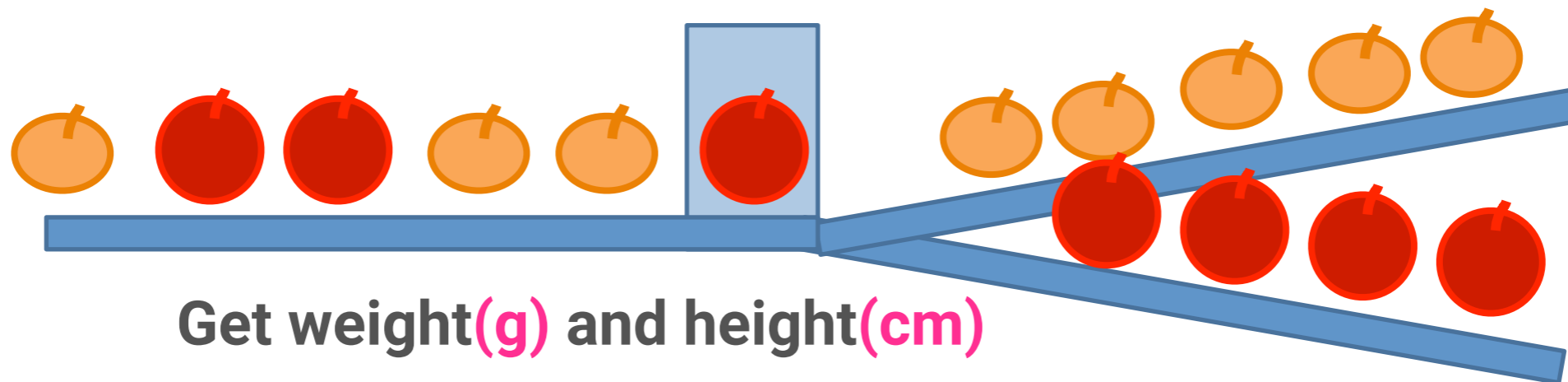
# Underlying principle: Use of statistical trends

**We see common patterns (empirical rules) emerge from observing many examples, which we cannot recognize when we see only a few.**

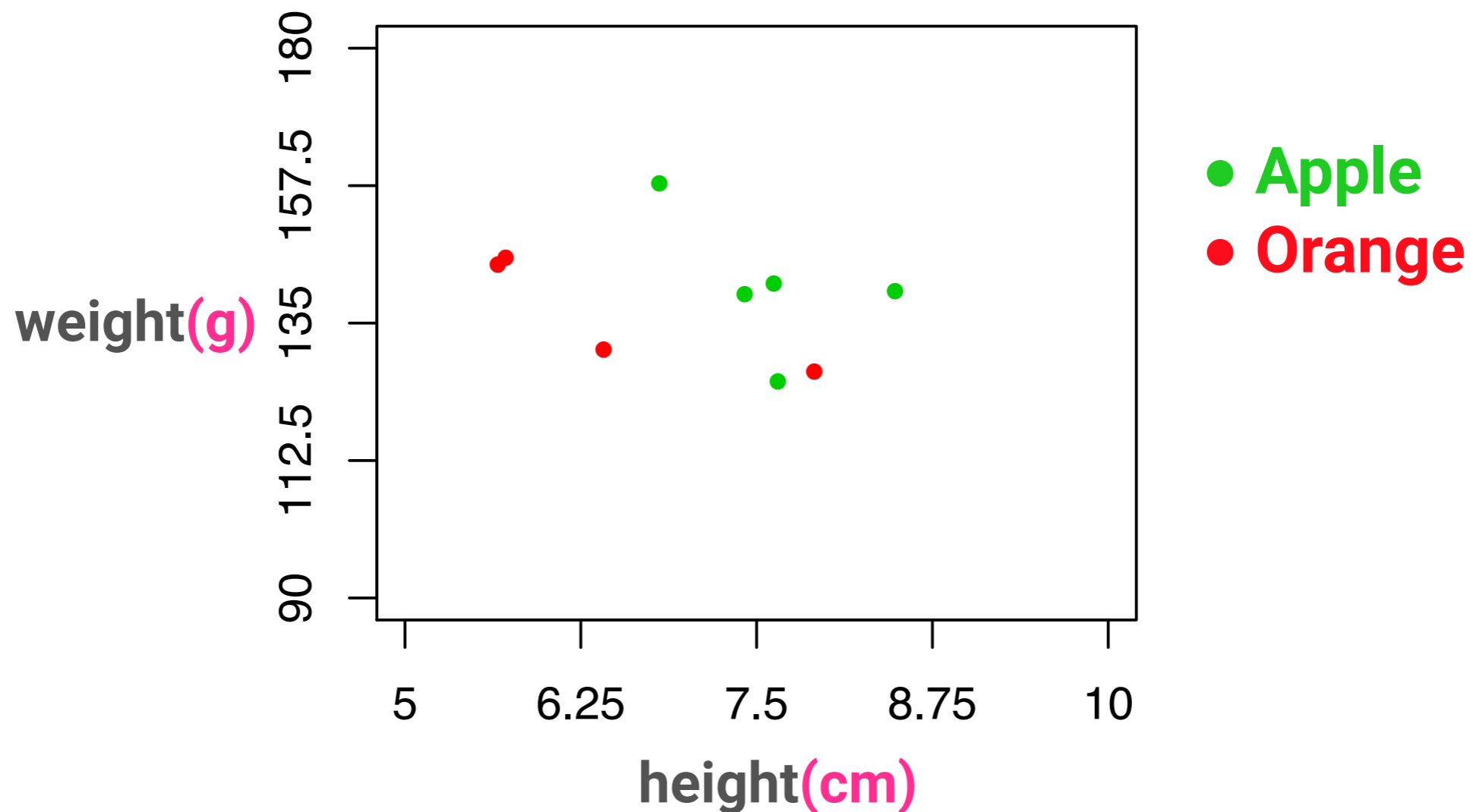
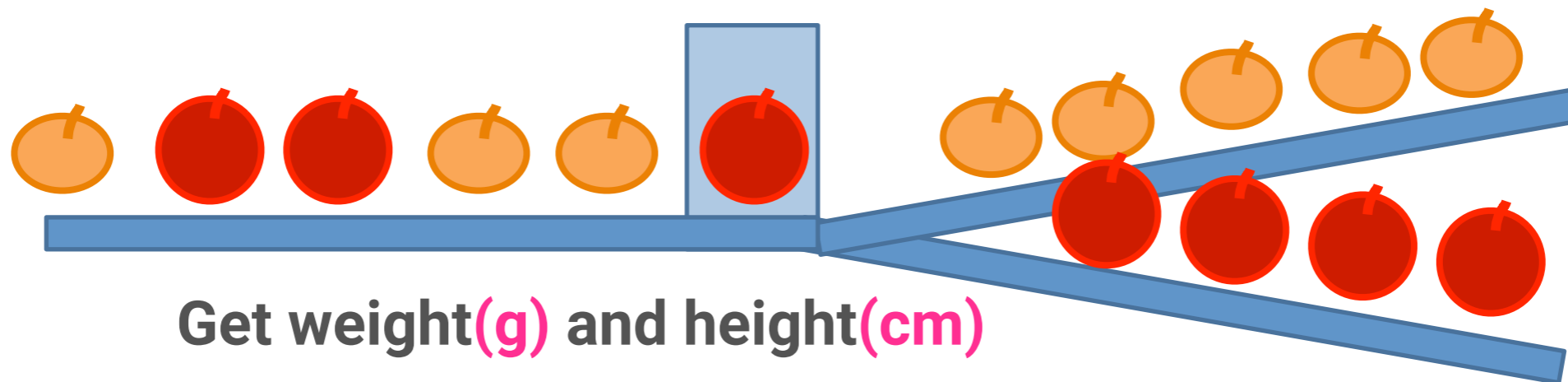


**Characterize the difference between “あ” and “め” not by explicit rules, but by implicit statistical rules directly defined by many observations.**

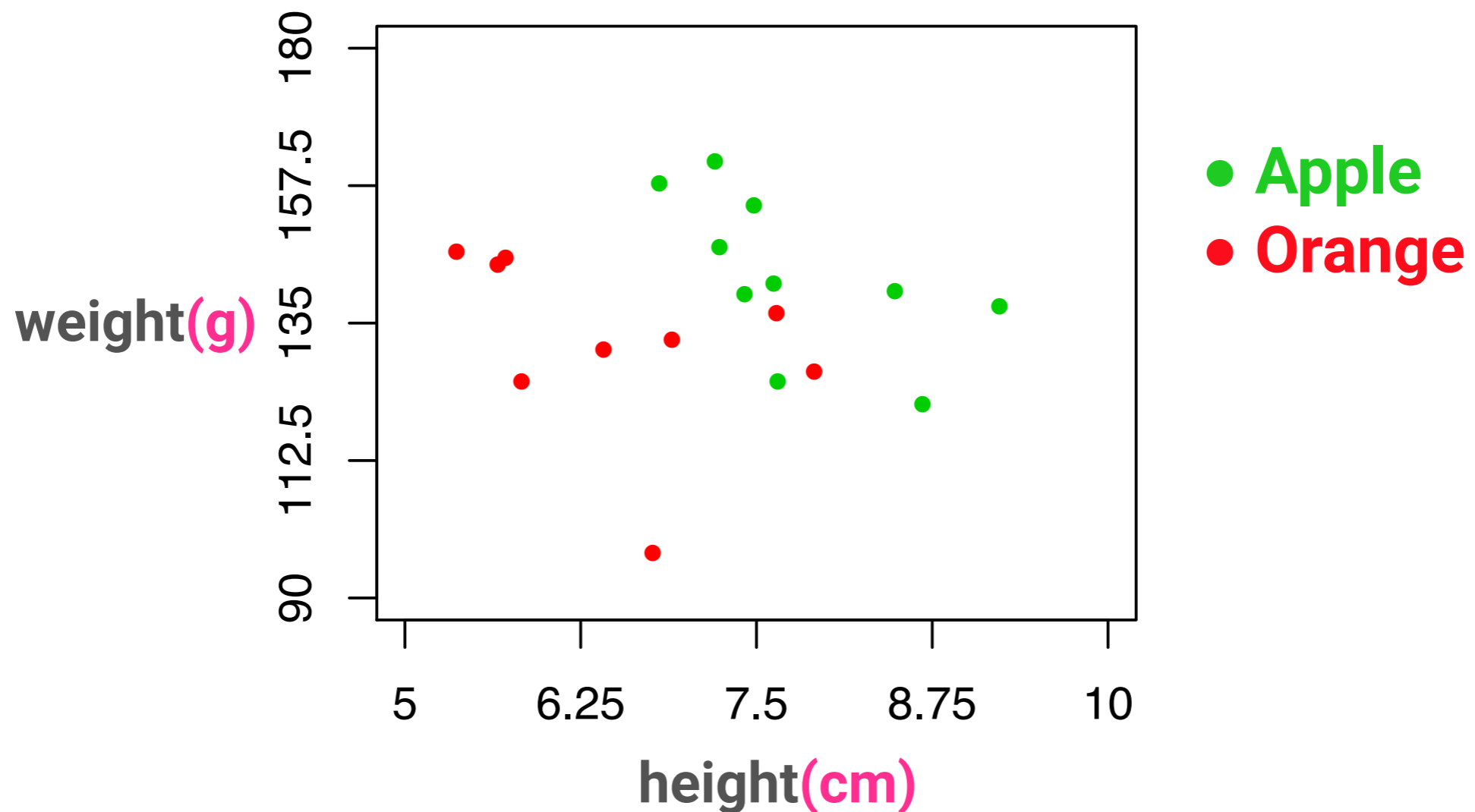
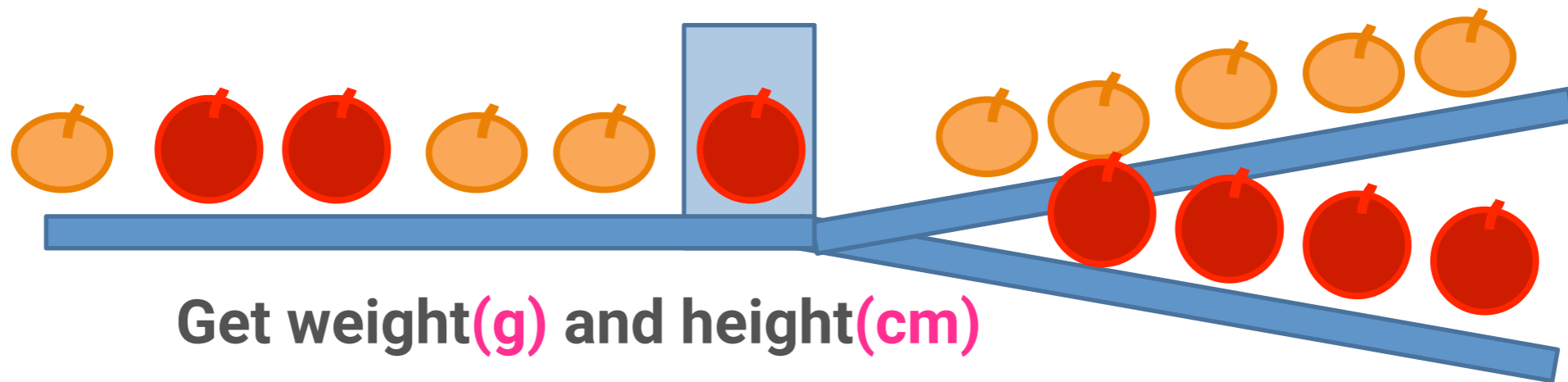
# Patterns emerge from many examples?



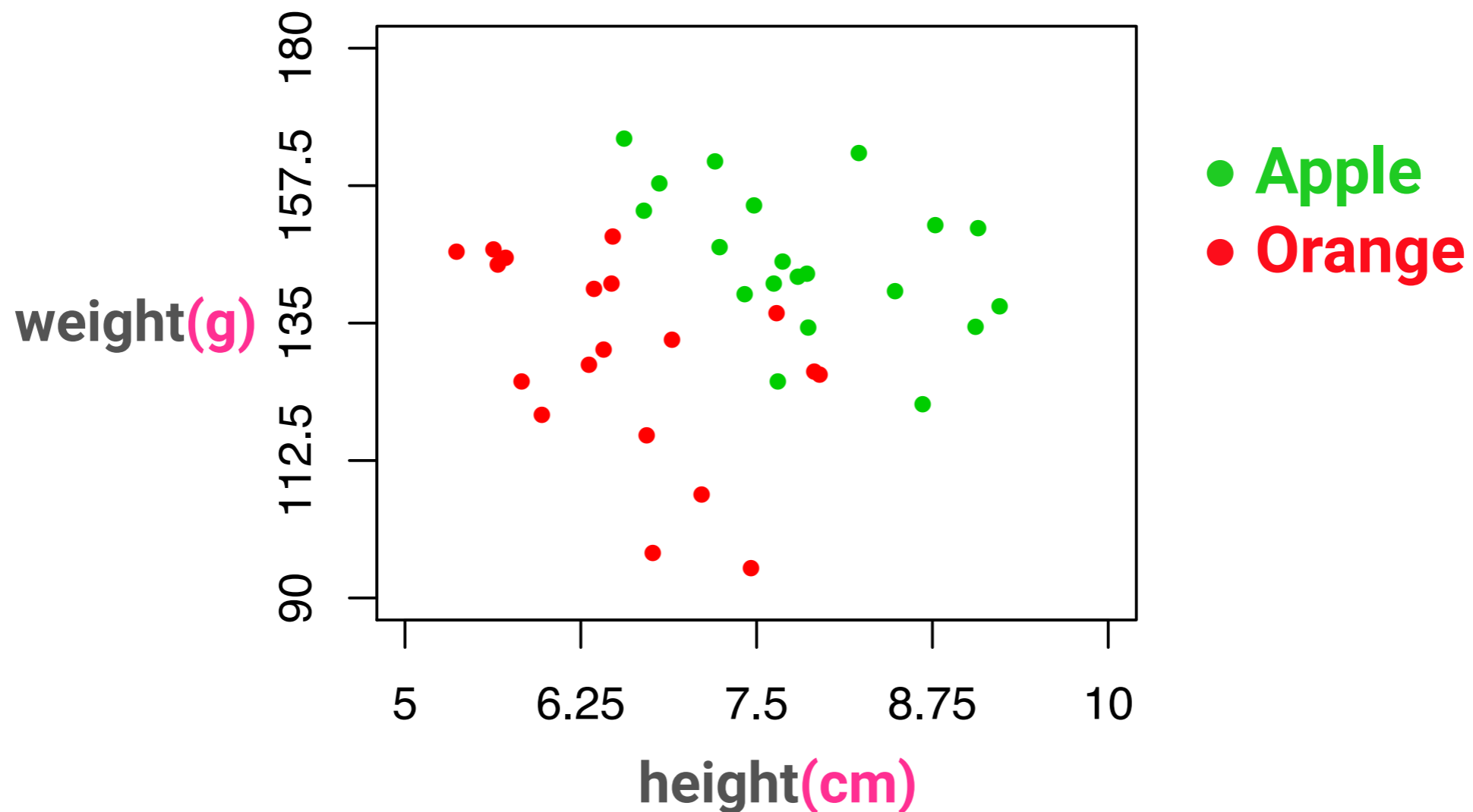
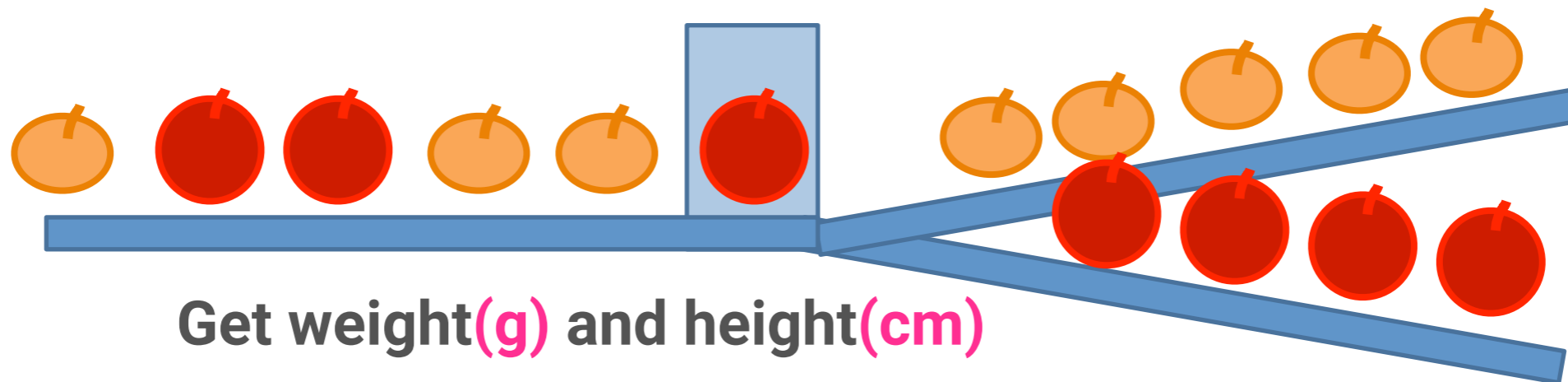
# Patterns emerge from many examples?



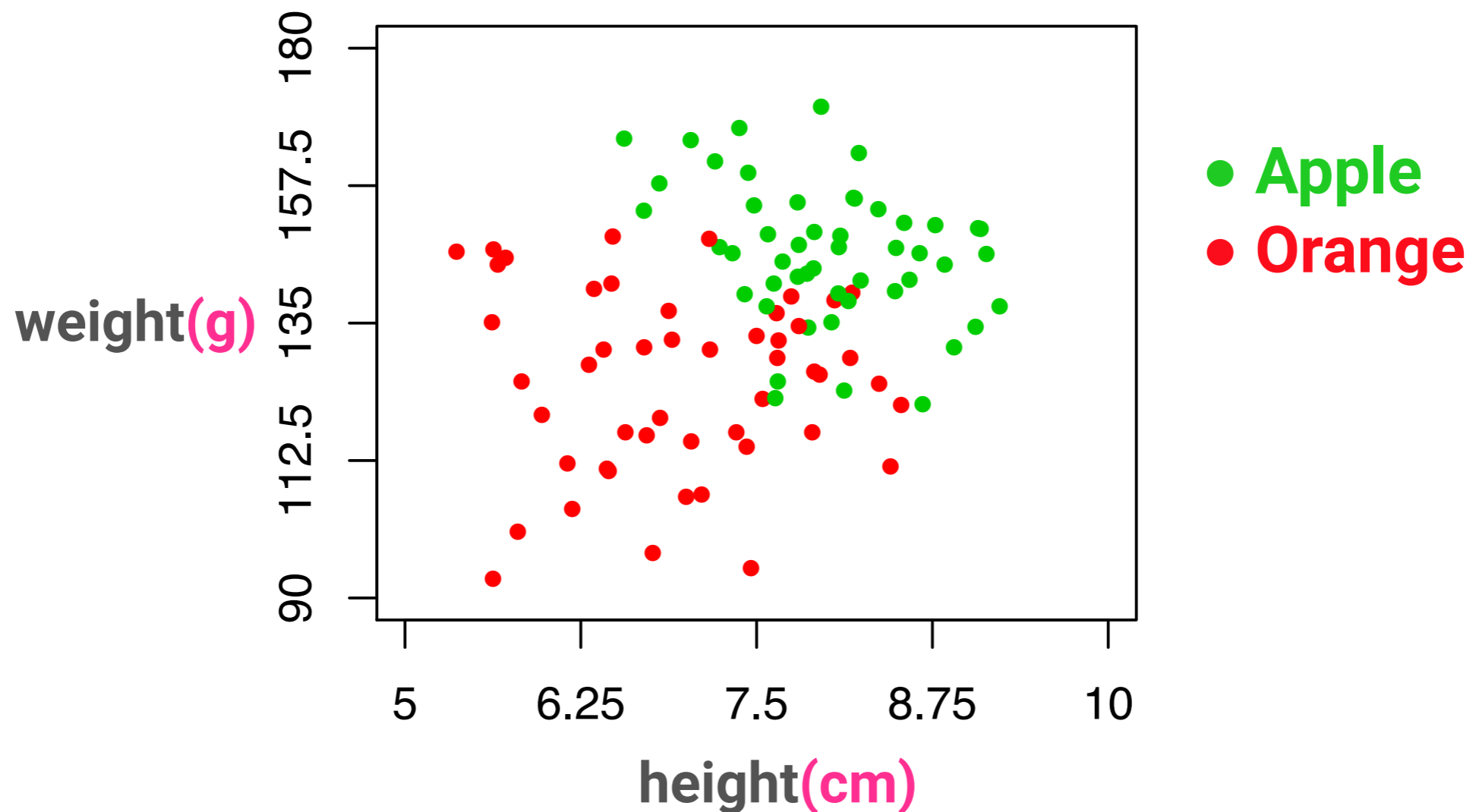
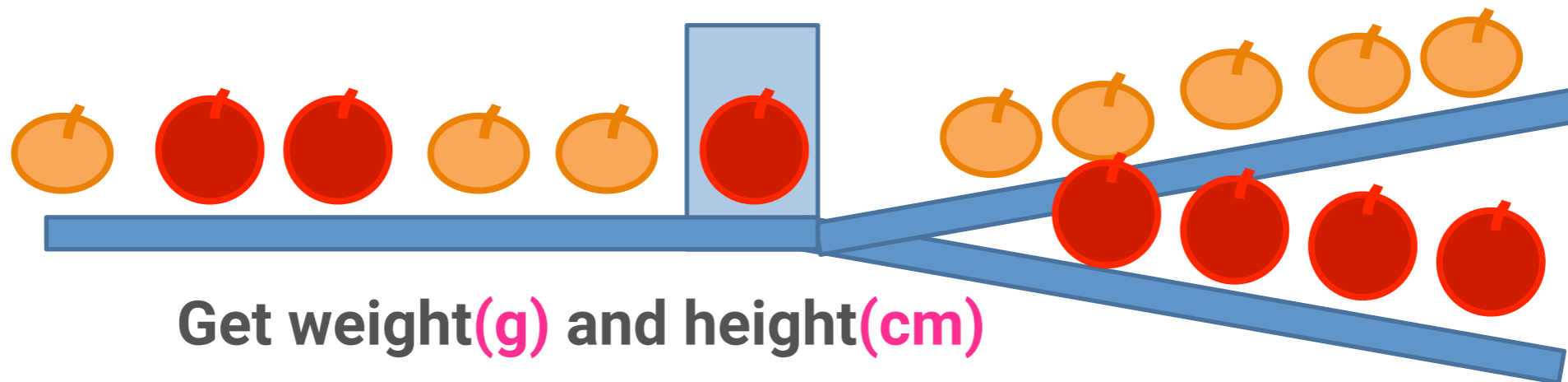
# Patterns emerge from many examples?



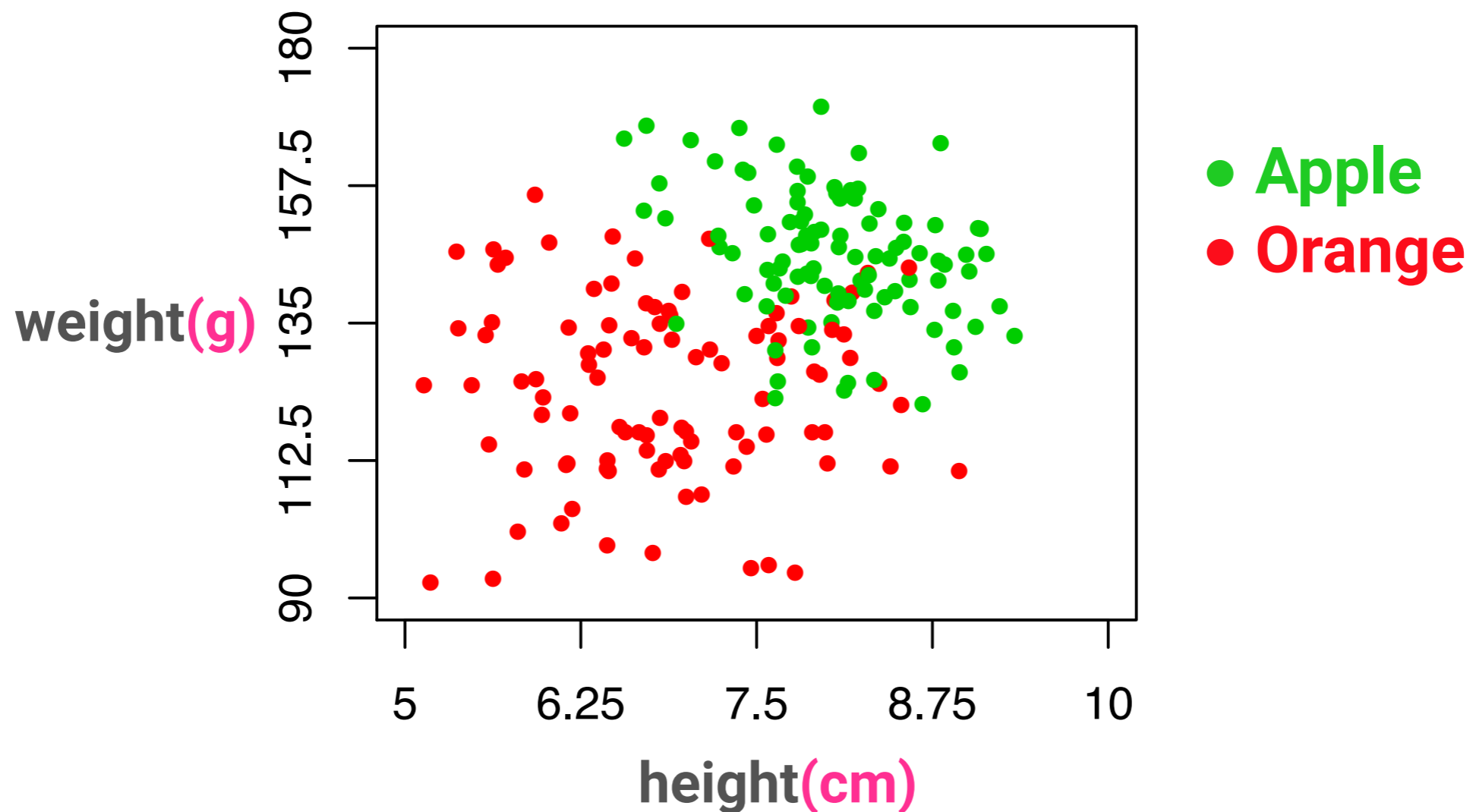
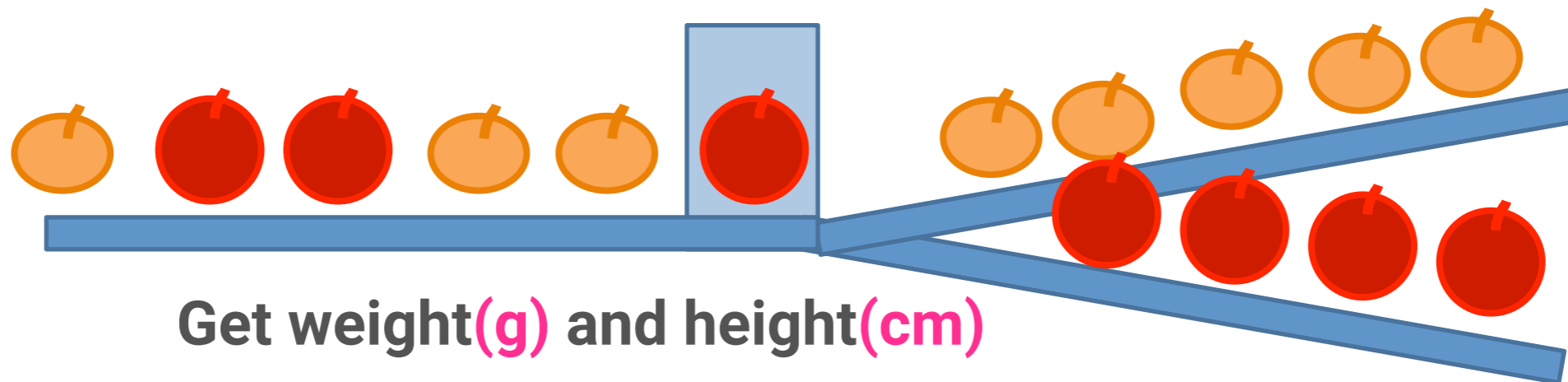
# Patterns emerge from many examples?



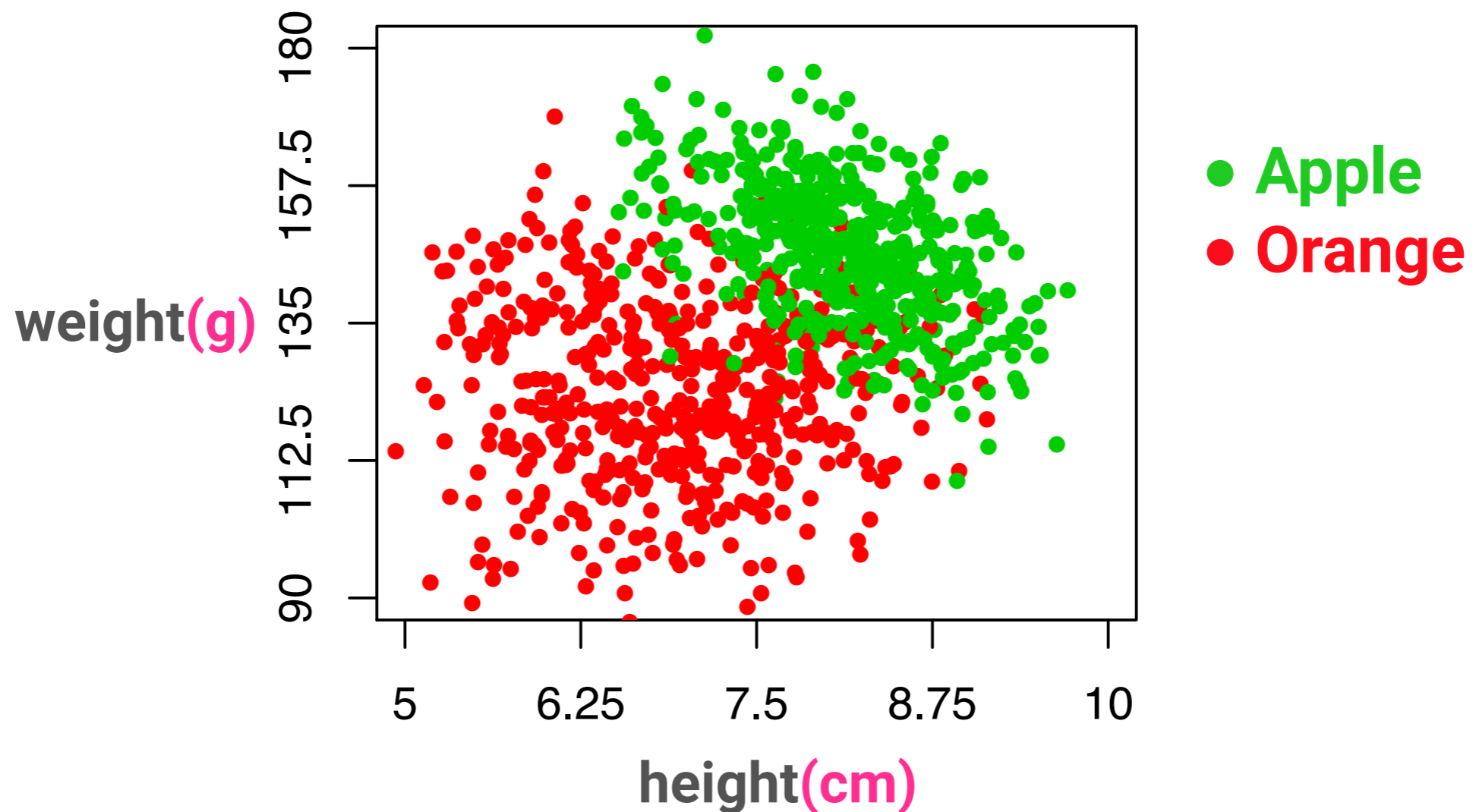
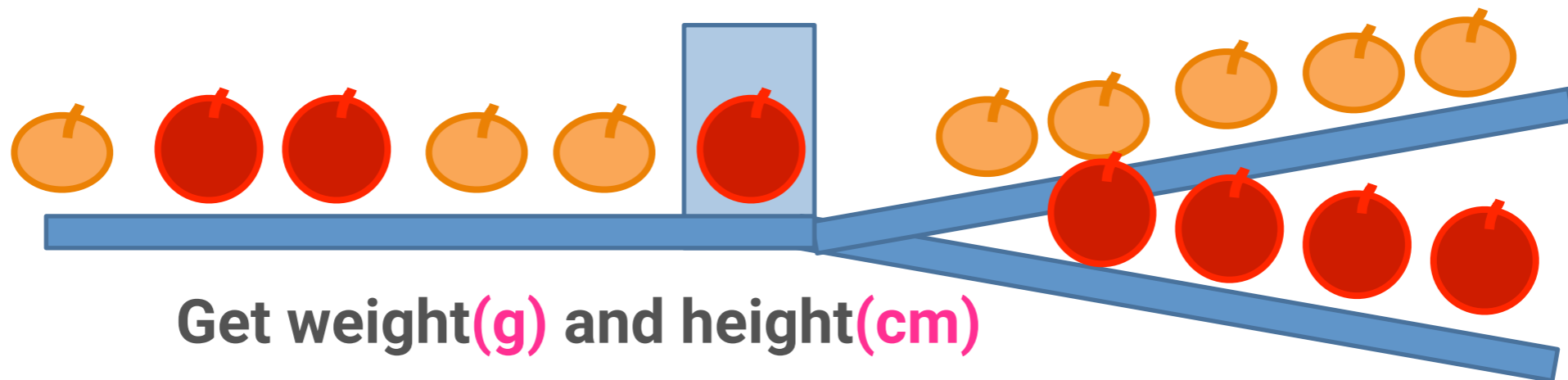
# Patterns emerge from many examples?



# Patterns emerge from many examples?

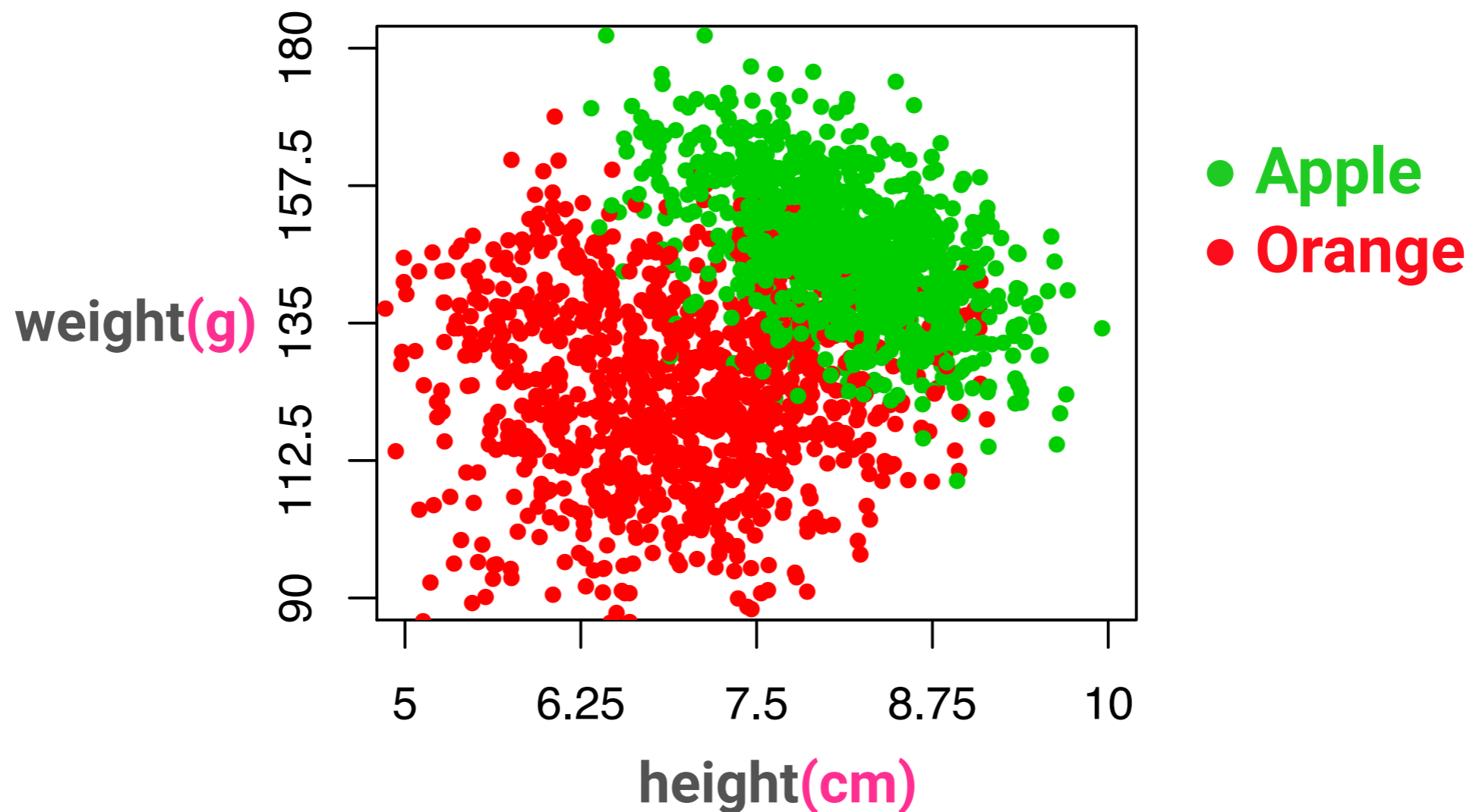
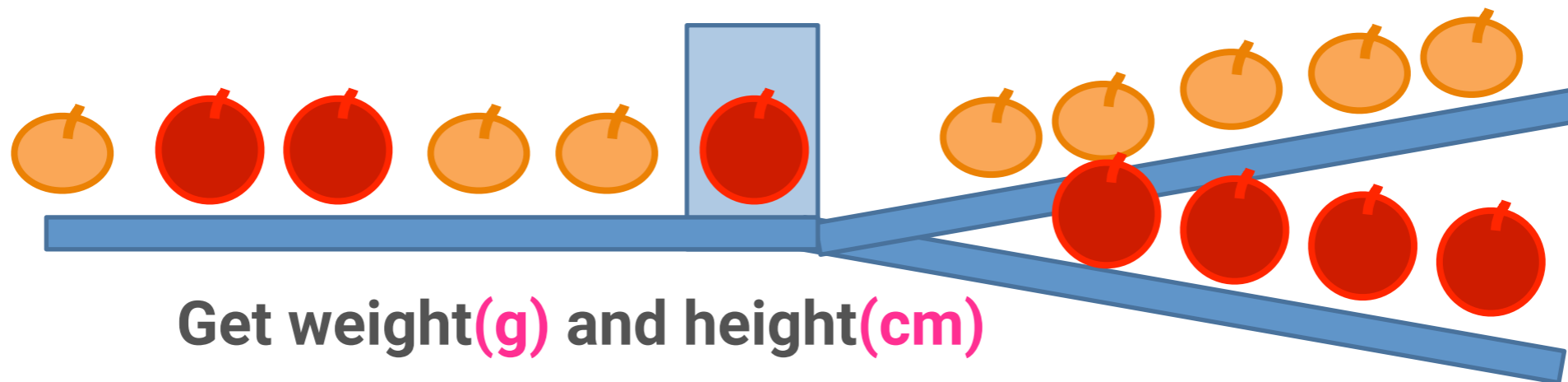


# Patterns emerge from many examples?

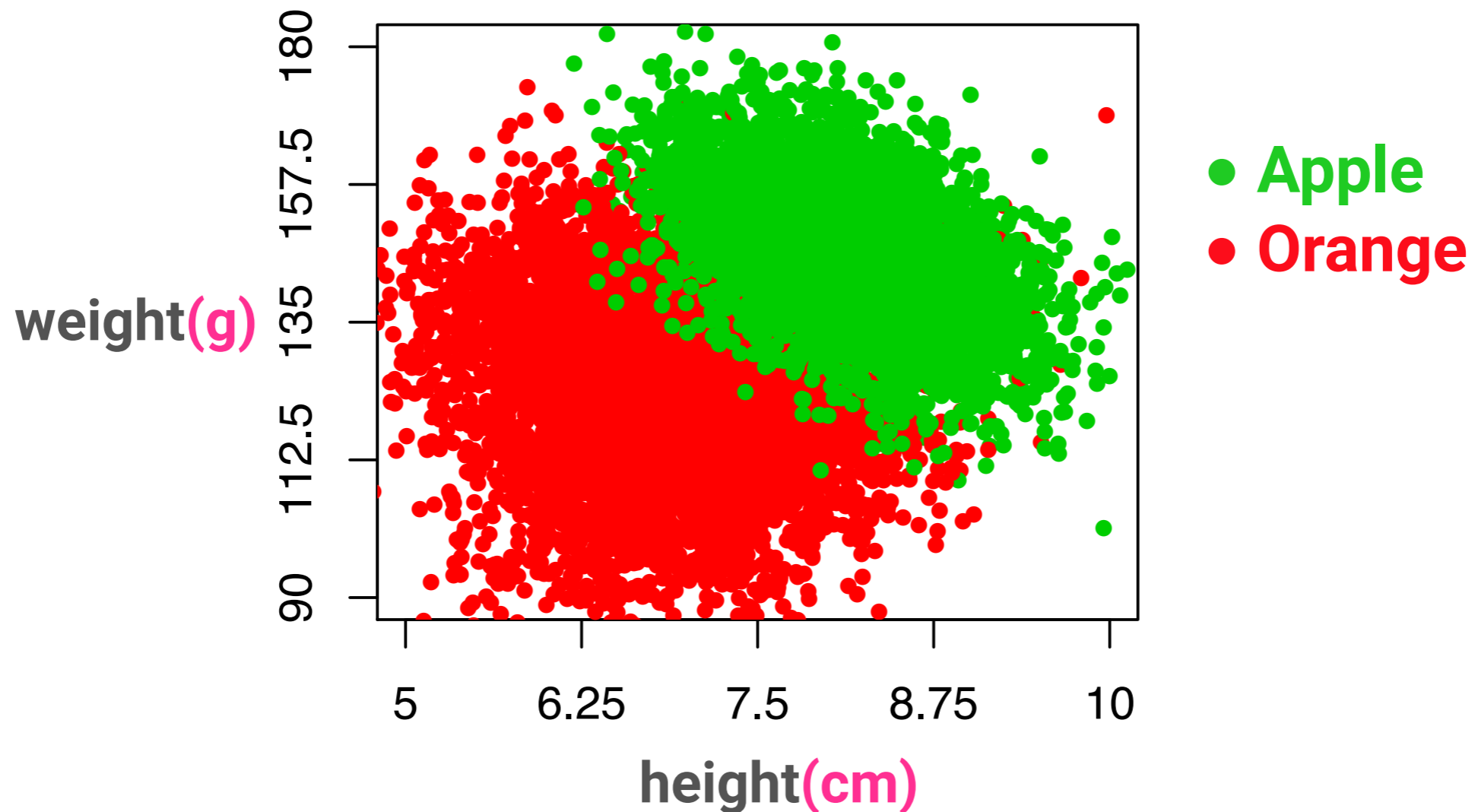
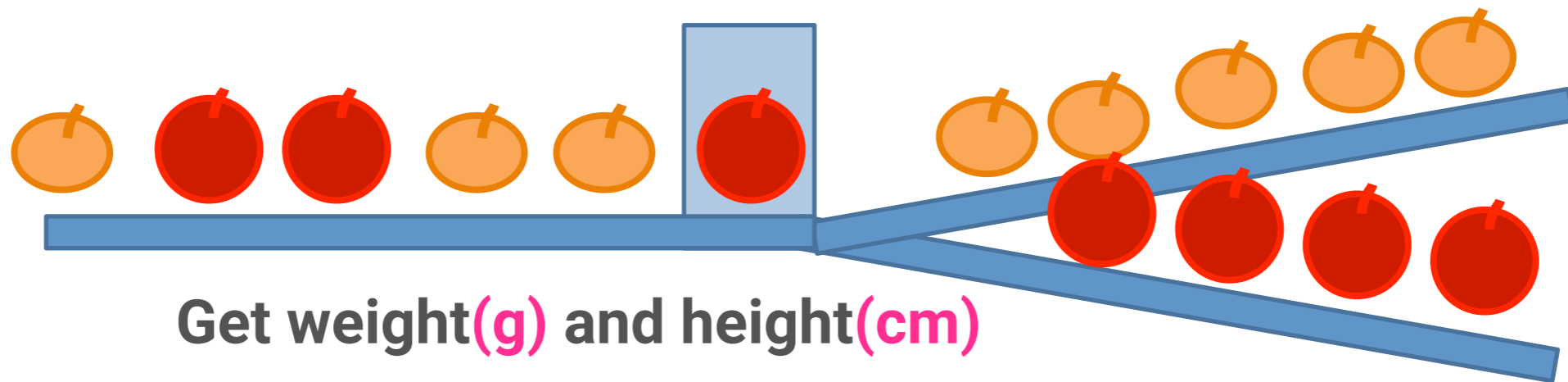




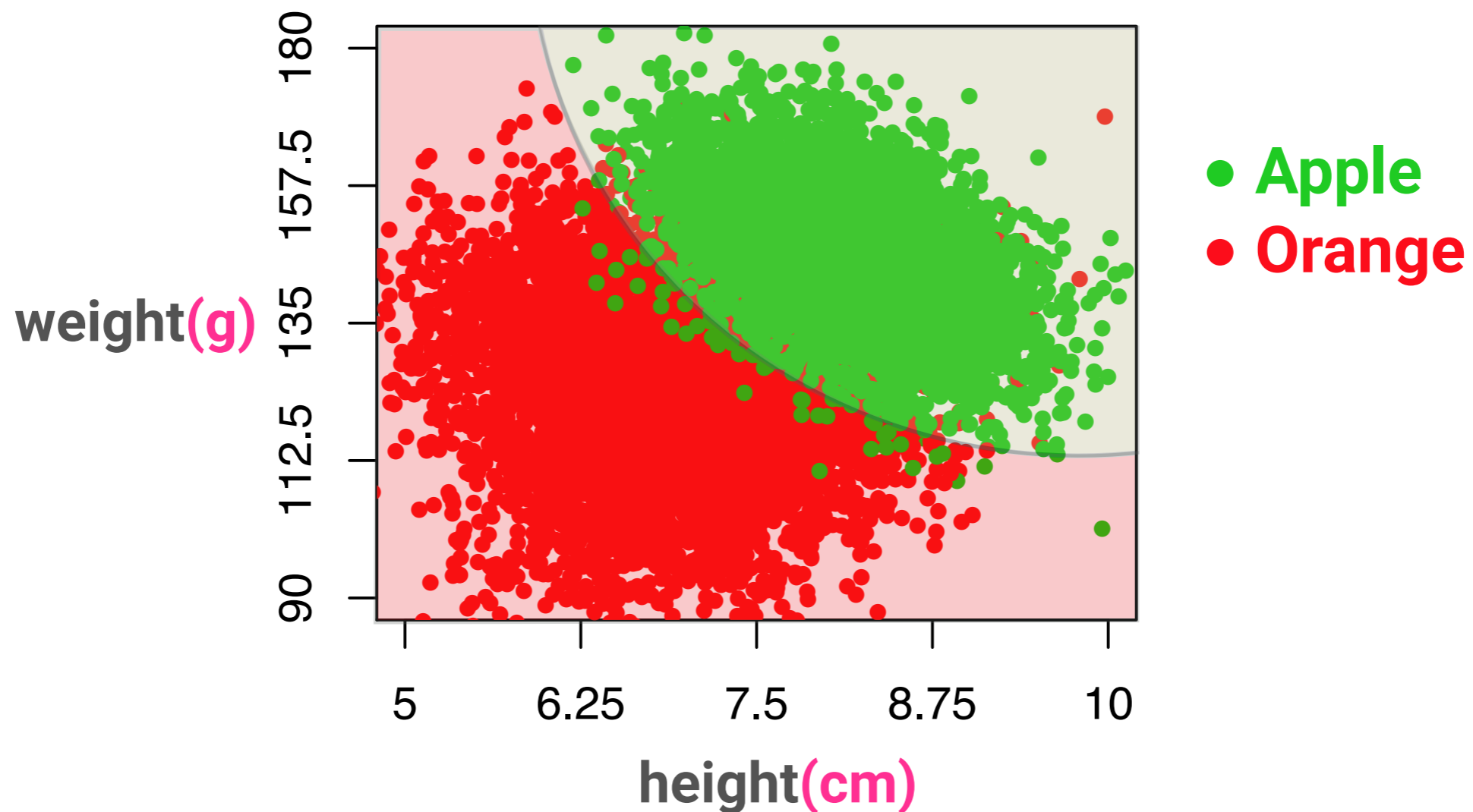
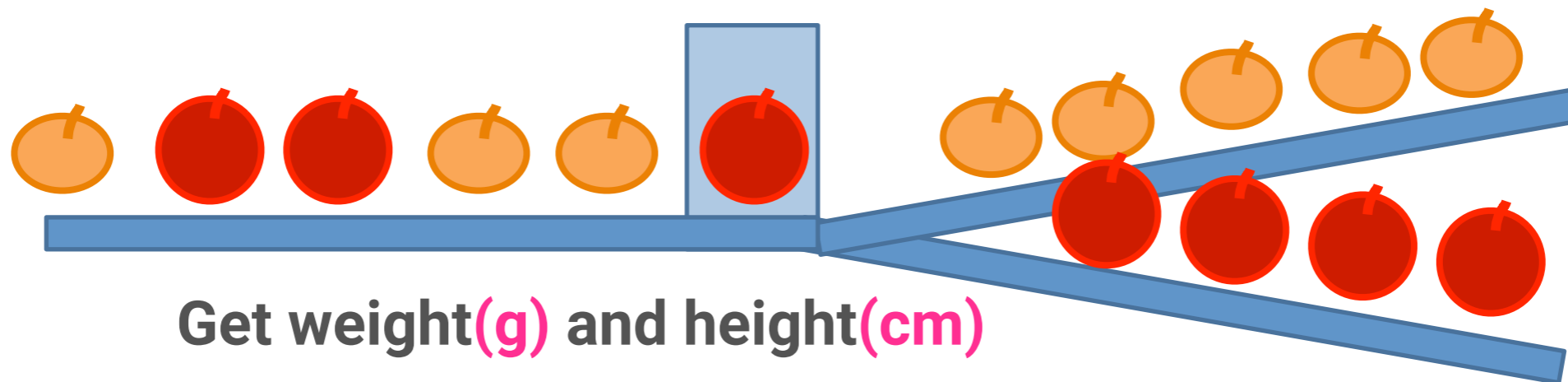
# Patterns emerge from many examples?



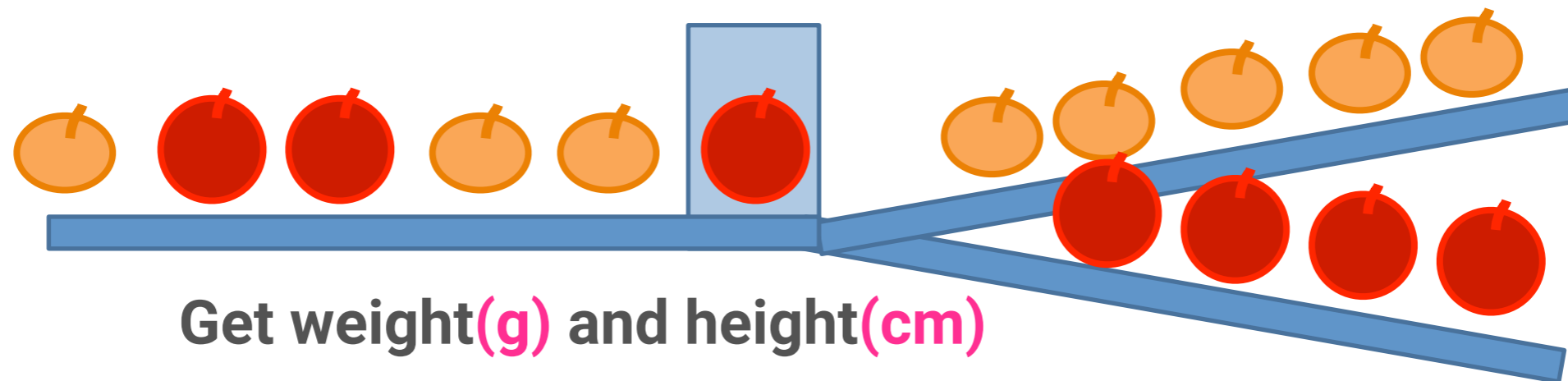
# Patterns emerge from many examples?



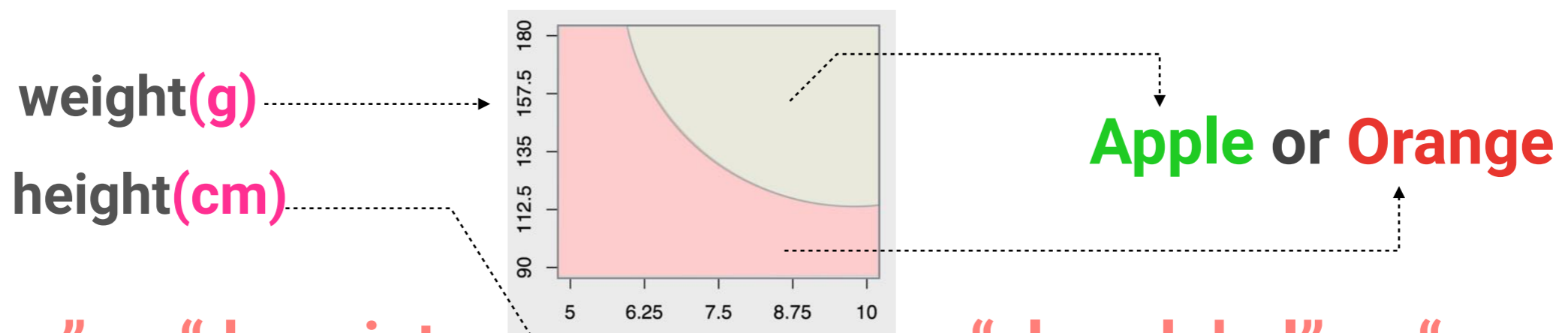
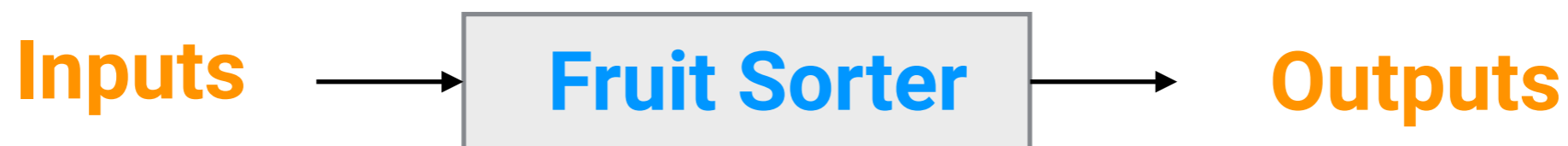
# Patterns emerge from many examples?



# This is all about (supervised) machine learning



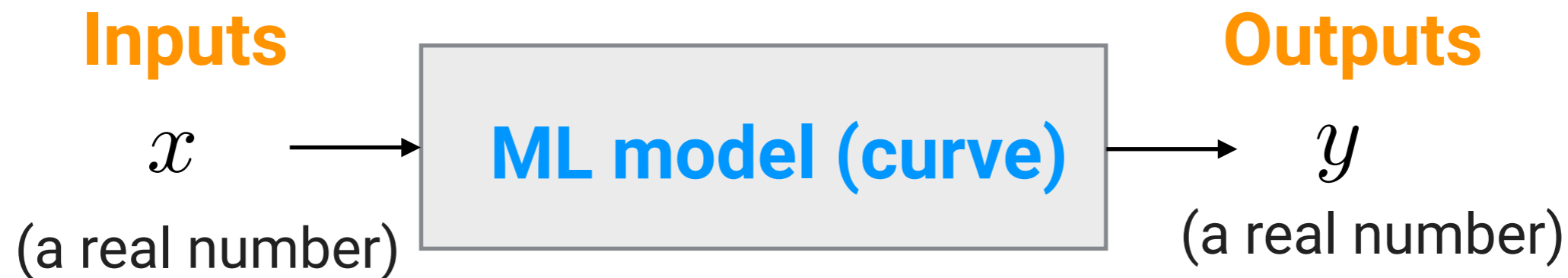
Now we got **a computer program** for this classification problem.



“features” or “descriptors”

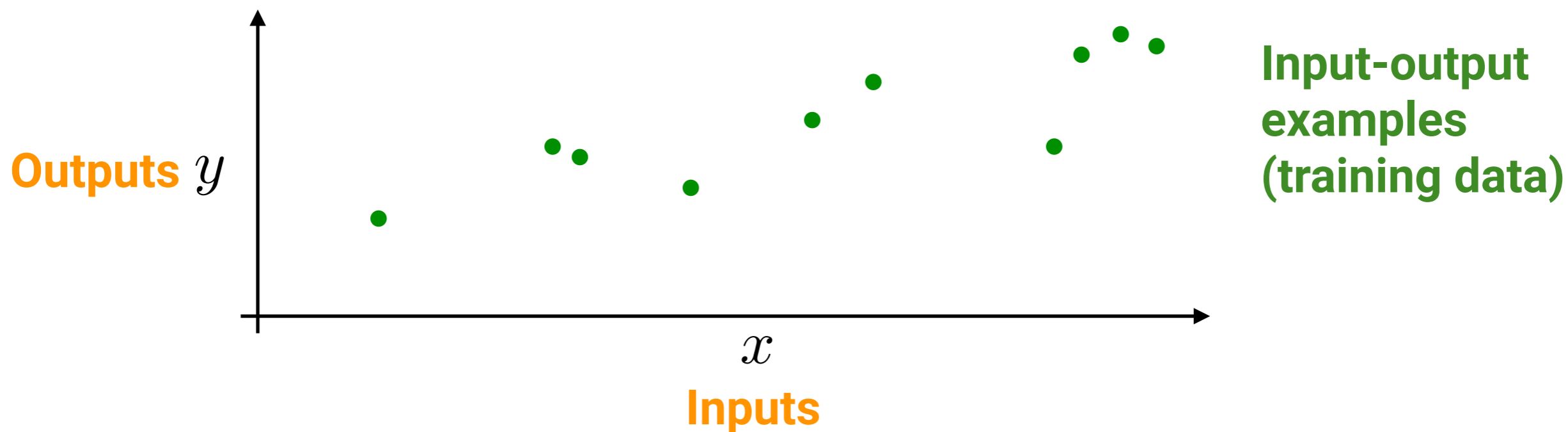
“class label” or “response”

# 1-dimensional real-number cases = curve fitting

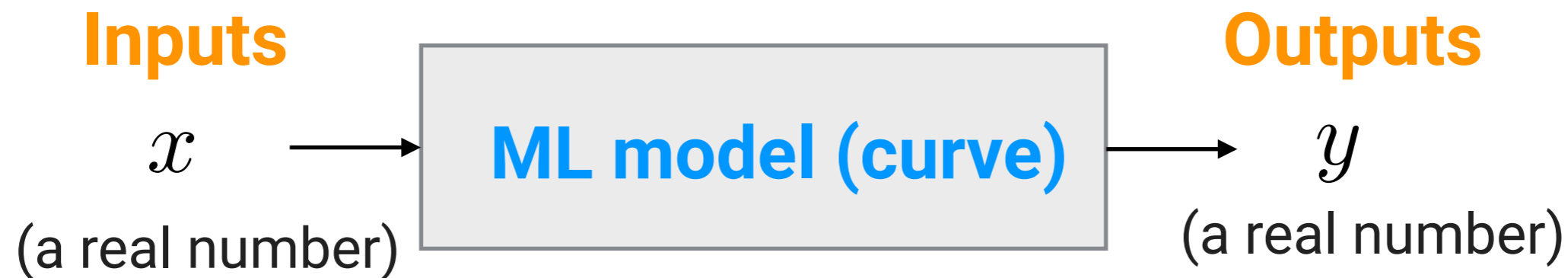


**Training data**  
( $n$  examples)

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$$

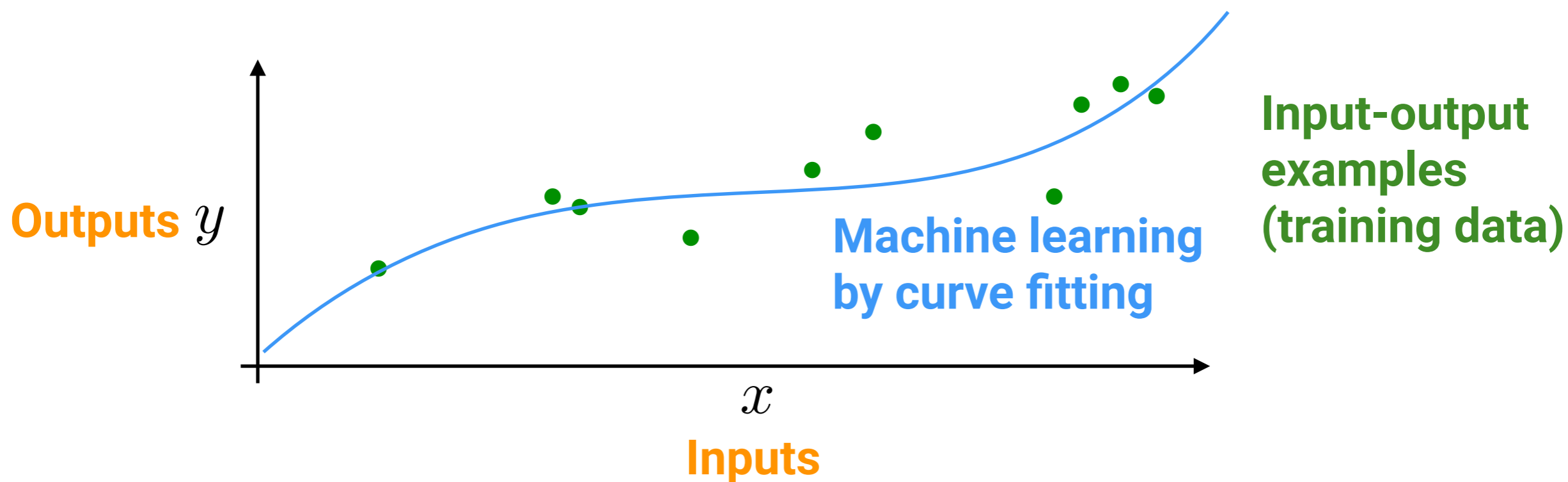


# 1-dimensional real-number cases = curve fitting

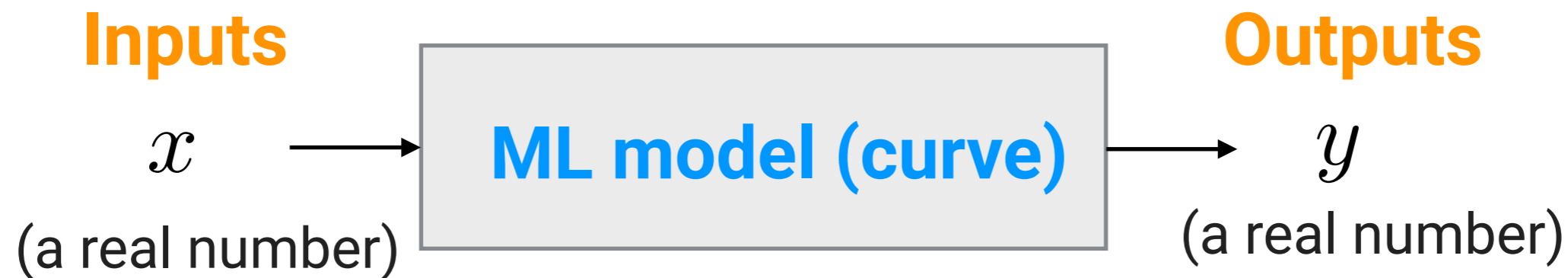


**Training data**  
( $n$  examples)

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$$

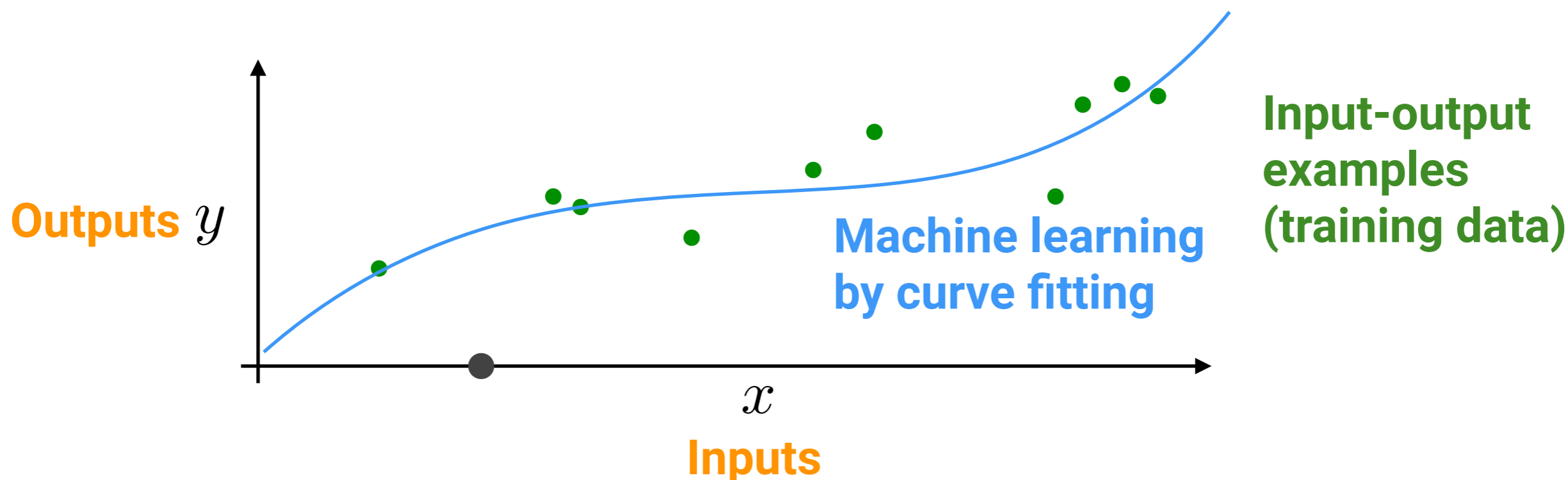


# 1-dimensional real-number cases = curve fitting

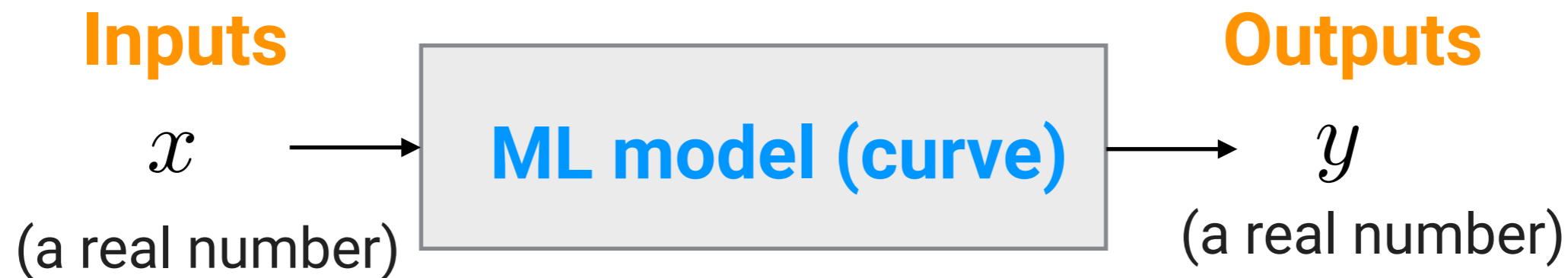


**Training data**  
( $n$  examples)

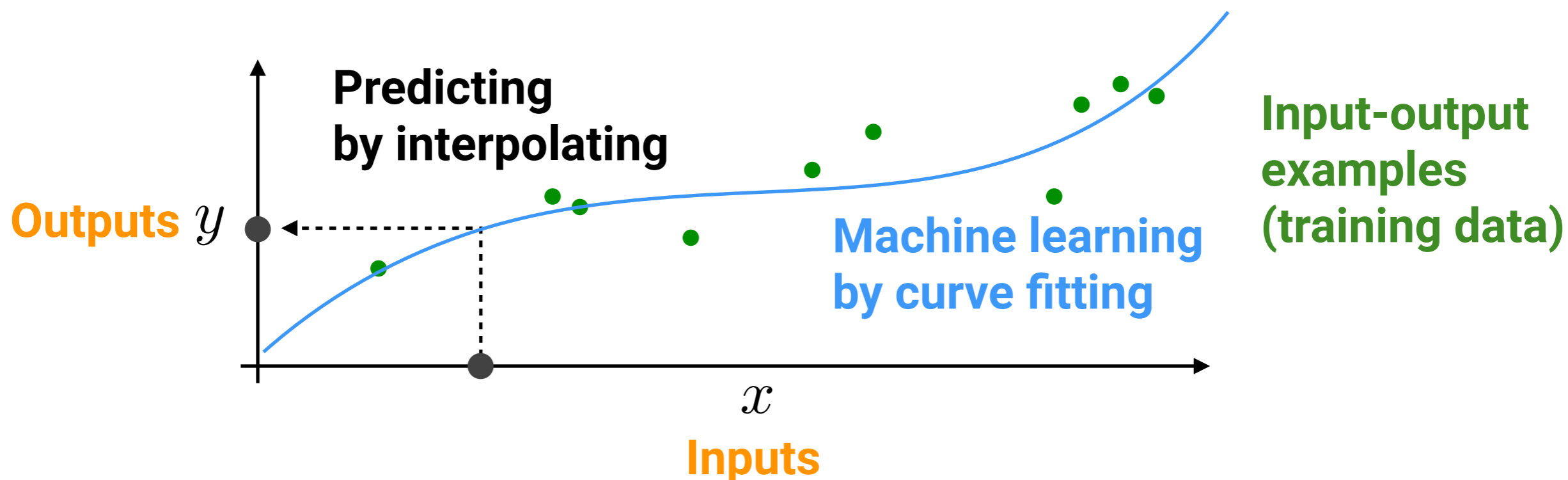
$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$$



# 1-dimensional real-number cases = curve fitting

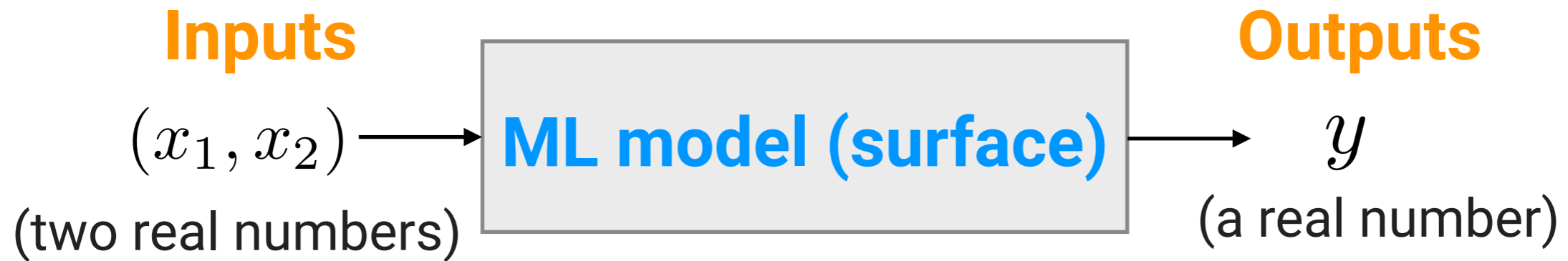


**Training data** ( $n$  examples)  $\{ (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)}) \}$



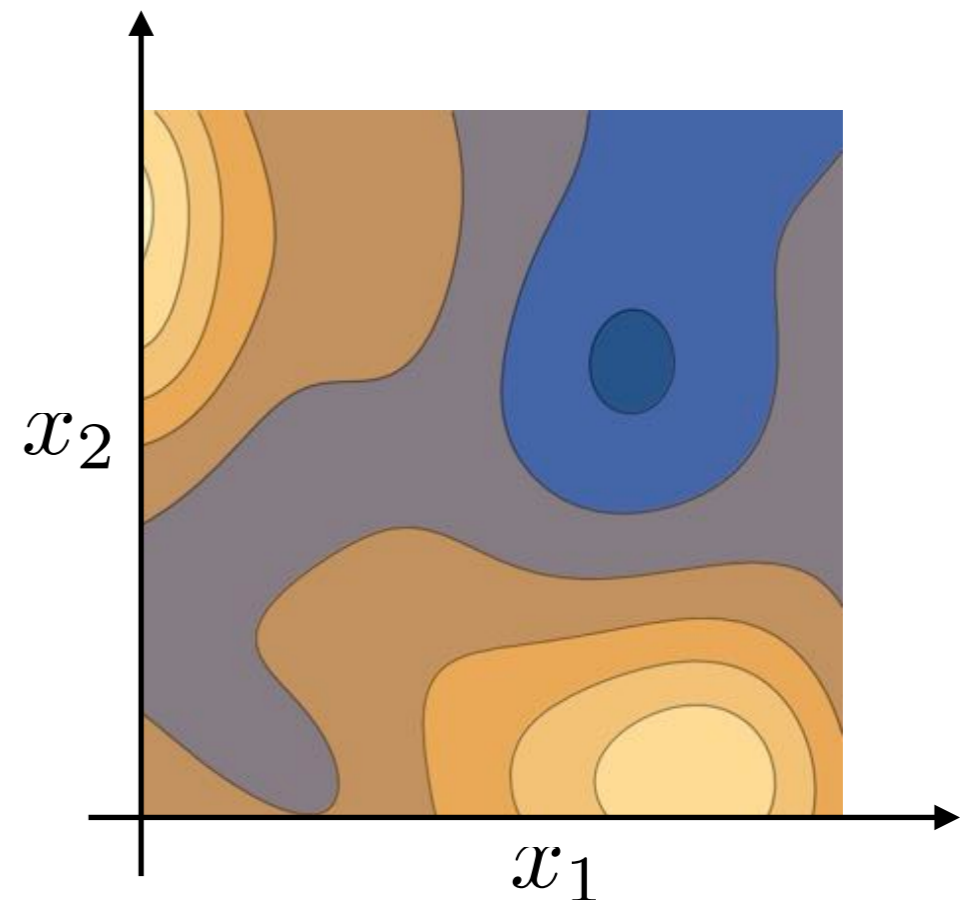
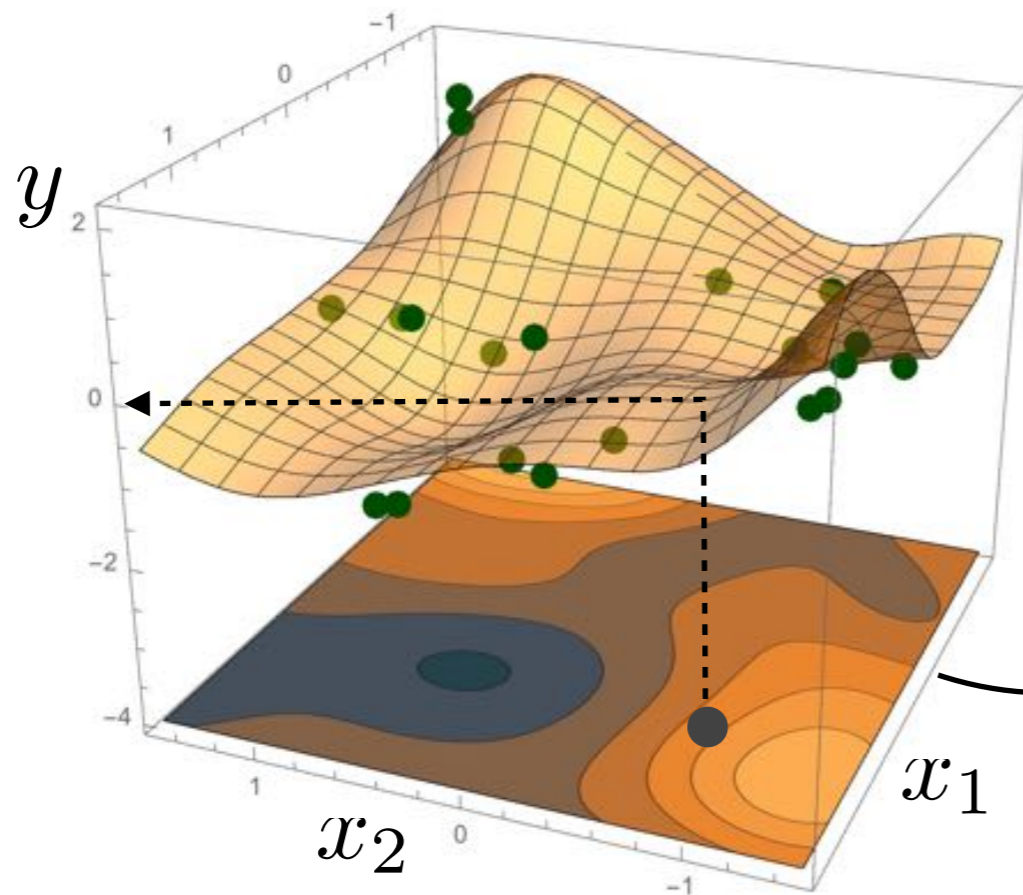


# 2-dimensional real-number cases = surface fitting



**Training data**  
( $n$  examples)

$$\left\{ \left( (x_1^{(1)}, x_2^{(1)}), y^{(1)} \right), \dots, \left( (x_1^{(n)}, x_2^{(n)}), y^{(n)} \right) \right\}$$



# High-dimensional cases = function fitting

**Inputs**

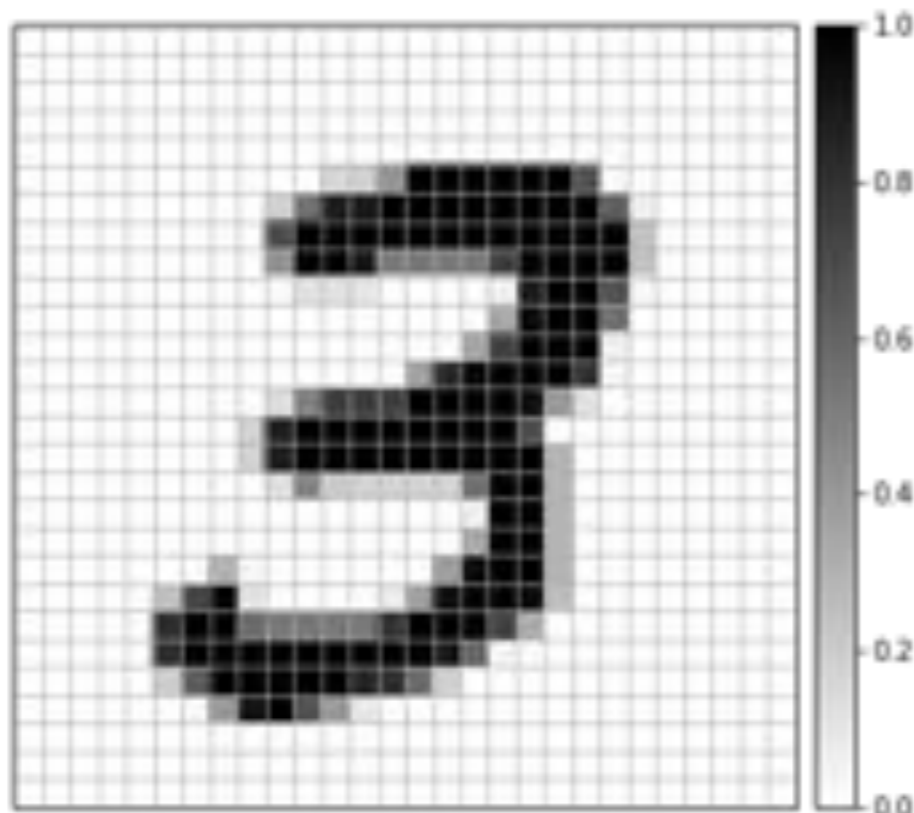
$(x_1, x_2, \dots, x_{784})$   
(784 real numbers)

**ML model (surface)**

**Outputs**

$y$   
(10 real numbers)

An 28 x 28 (=784) pixel image



Probabilities for 0,1,2,...,9

0	1	2	3	4
0.0	0.0	0.0	<b>0.9</b>	0.0
5	6	7	8	9
0.0	0.0	0.0	<b>0.1</b>	0.0

**Just fit a 10-dimensional-valued function in 784-dimensional space!**

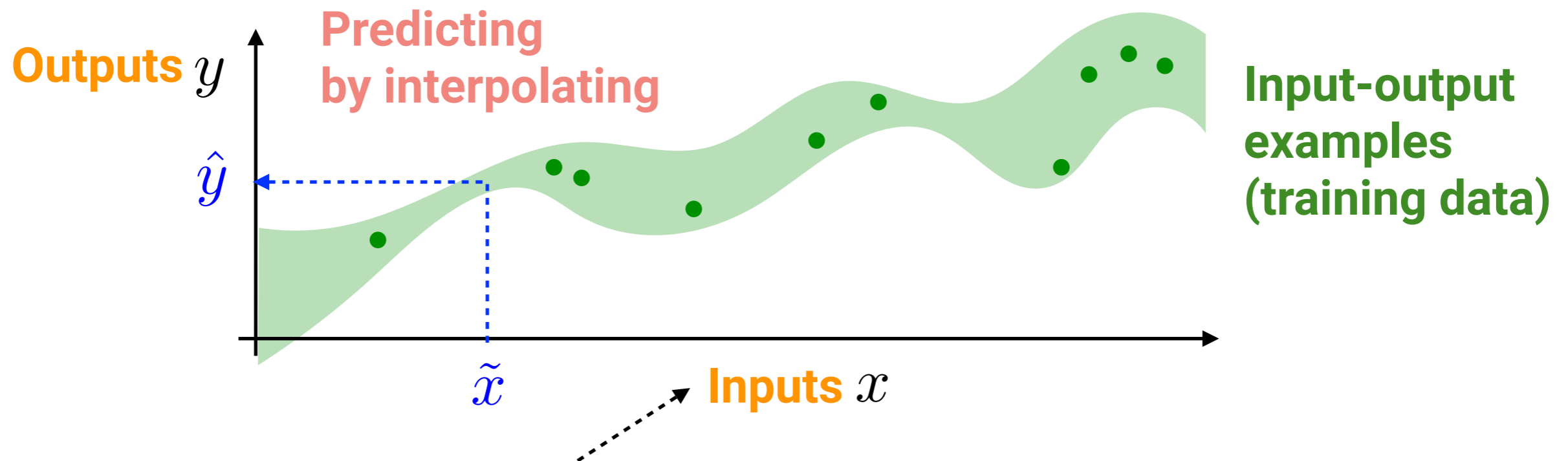
Find a nice mapping  $f : \mathbb{R}^{784} \rightarrow \mathbb{R}^{10}$

# Machine learning in a nutshell

What ML is doing to tell something valuable about “**data in the future**” from “**data we already have at hand**” is:

**High-dimensional interpolation by function fitting**

↳ any surface/curve model



This can be **very high-dimensional** rather than only 1 dimensional or so in practical situations.

# Machine learning in a nutshell

The current “machine learning” usually means

Just an interpolation by curve fitting! 😏

and does **NOT** means

- any human-like flexible and deep thinking/reasoning
- any magical ways to bring something unknowable

Aha! Just an interpolation? I know. It'll be easy! 😏

**Unfortunately NO.** First of all, curve fitting **in a high-dimensional space** is not trivial at all, technically speaking. 😓

And many other hard things come out ... 😓

# High-dimensional interpolation is counter-intuitive

**pix2pix**



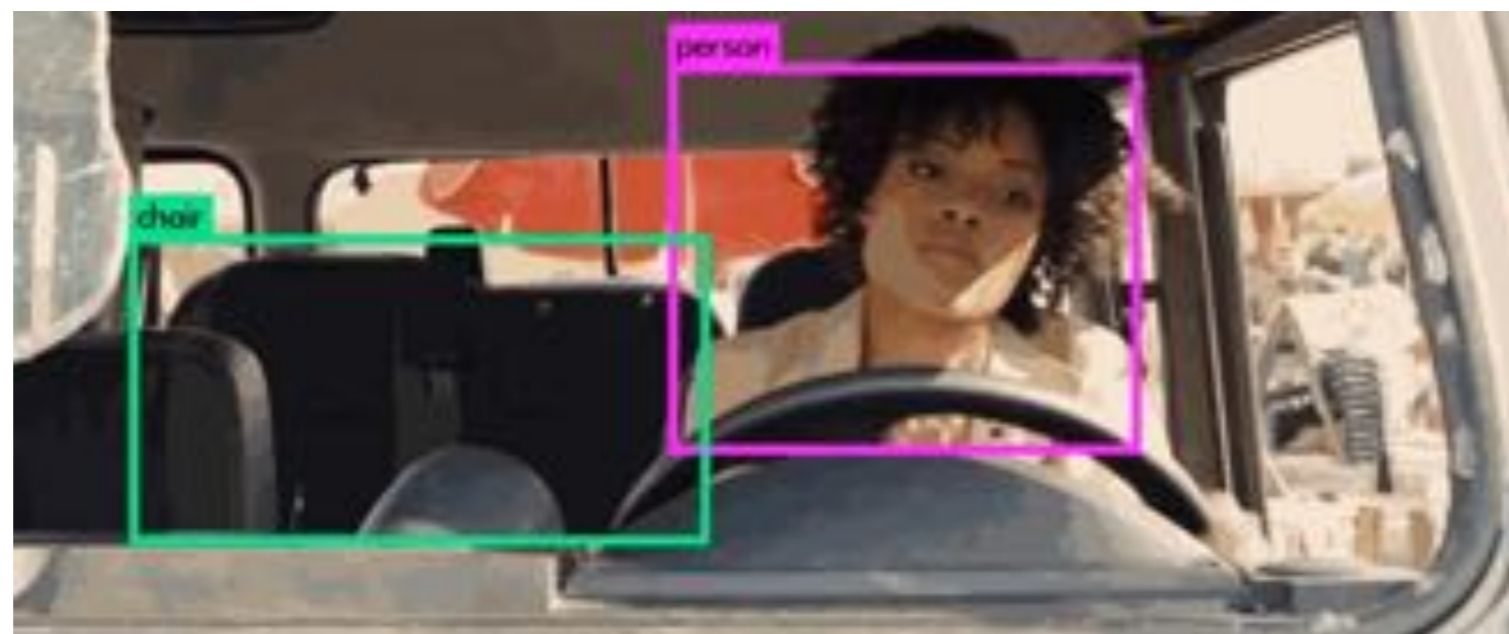
**Face swapping (e.g. DeepFake)**



**CycleGAN**



**YOLO**



# Even for machine-learning or AI experts!

Sales teams of high-tech companies sometimes set an unnecessarily high hurdle without knowing what is actually going on...

*IEEE Spectrum, 2019 Apr*

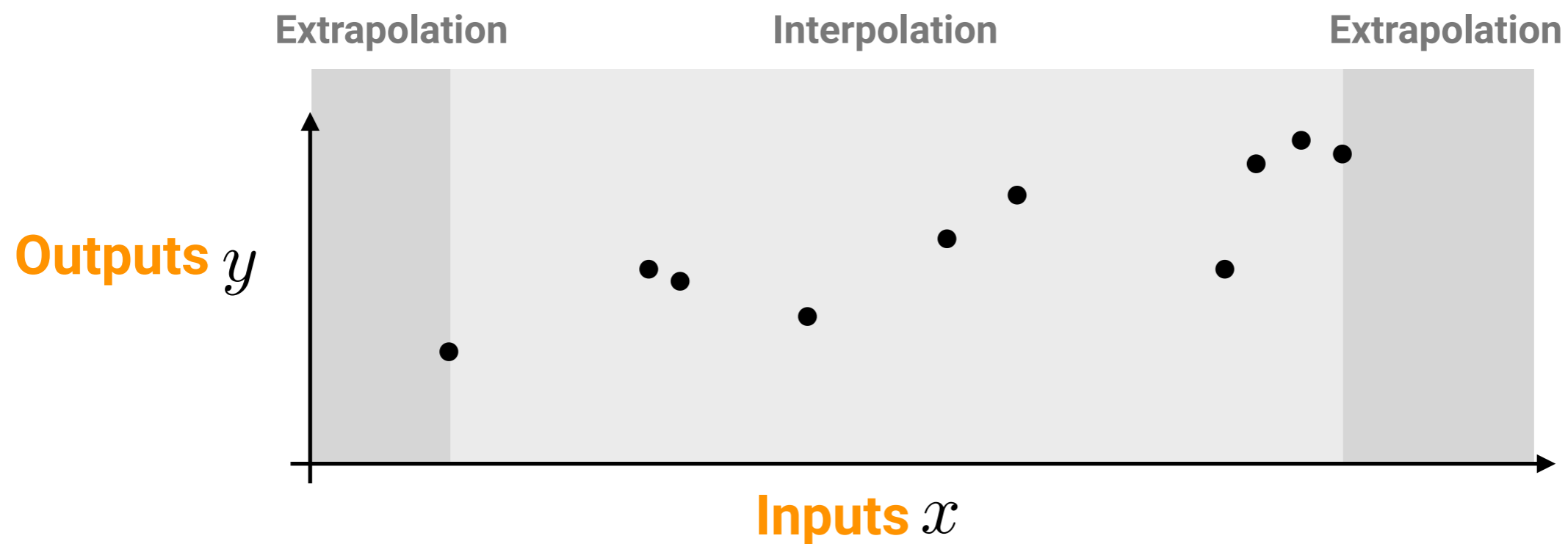
## IBM Watson, *Heal* Thyself

How IBM overpromised and  
underdelivered on AI health care

By ELIZA STRICKLAND ILLUSTRATIONS BY EDDIE GUY

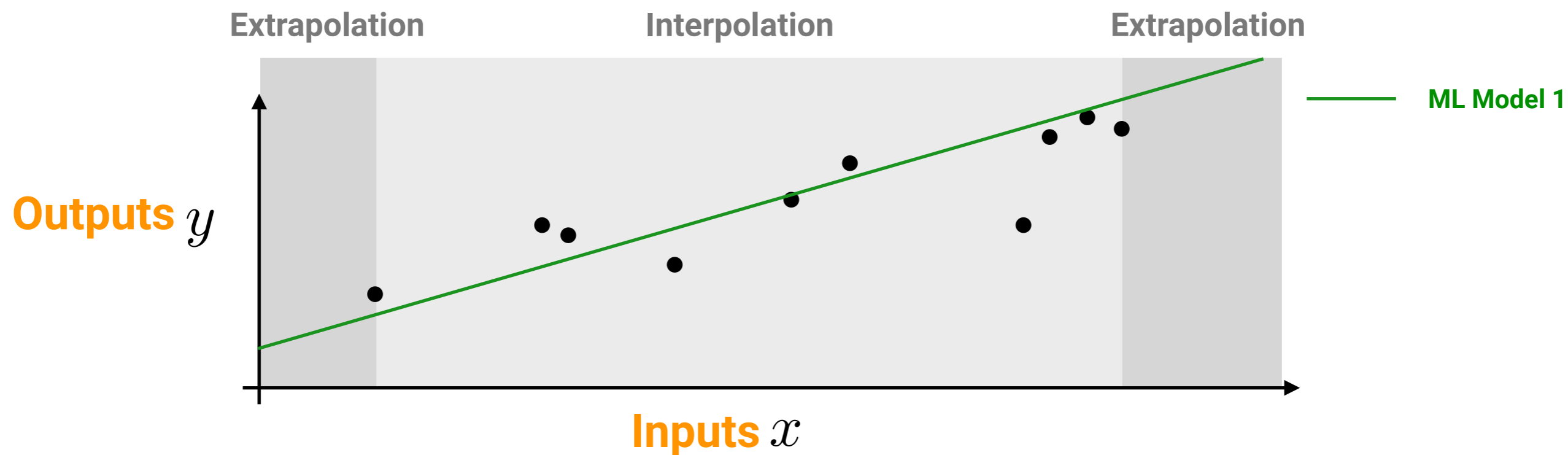


# Why ML looks so complicated?



(Generally high-dimensional rather than 1-dimensional)

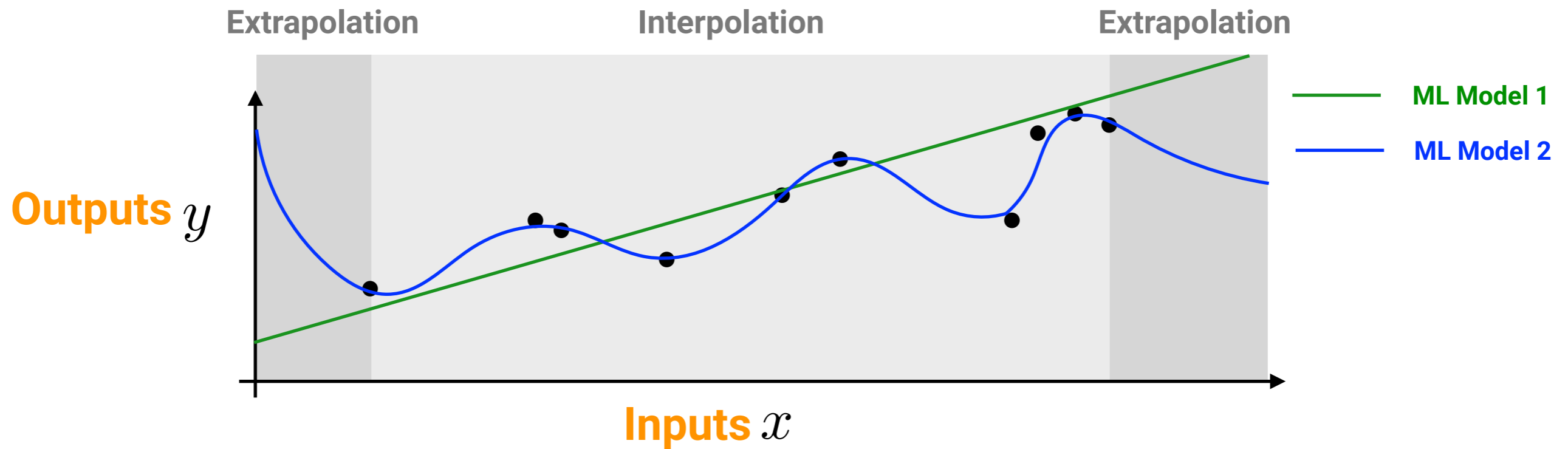
# Why ML looks so complicated?



(Generally high-dimensional rather than 1-dimensional)

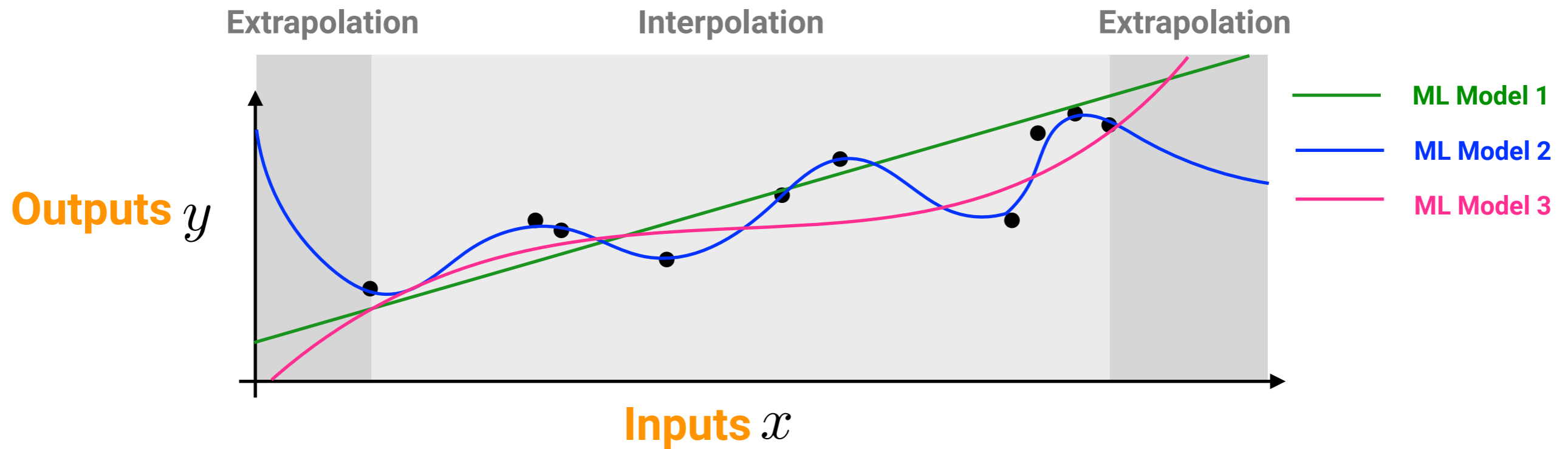


# Why ML looks so complicated?



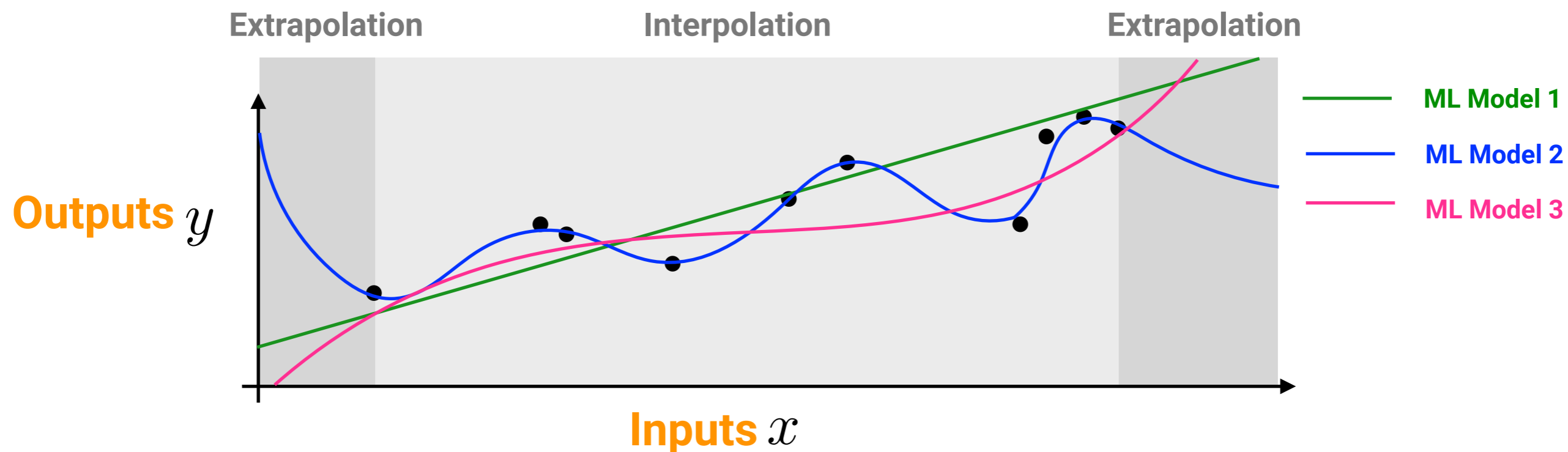
(Generally high-dimensional rather than 1-dimensional)

# Why ML looks so complicated?



(Generally high-dimensional rather than 1-dimensional)

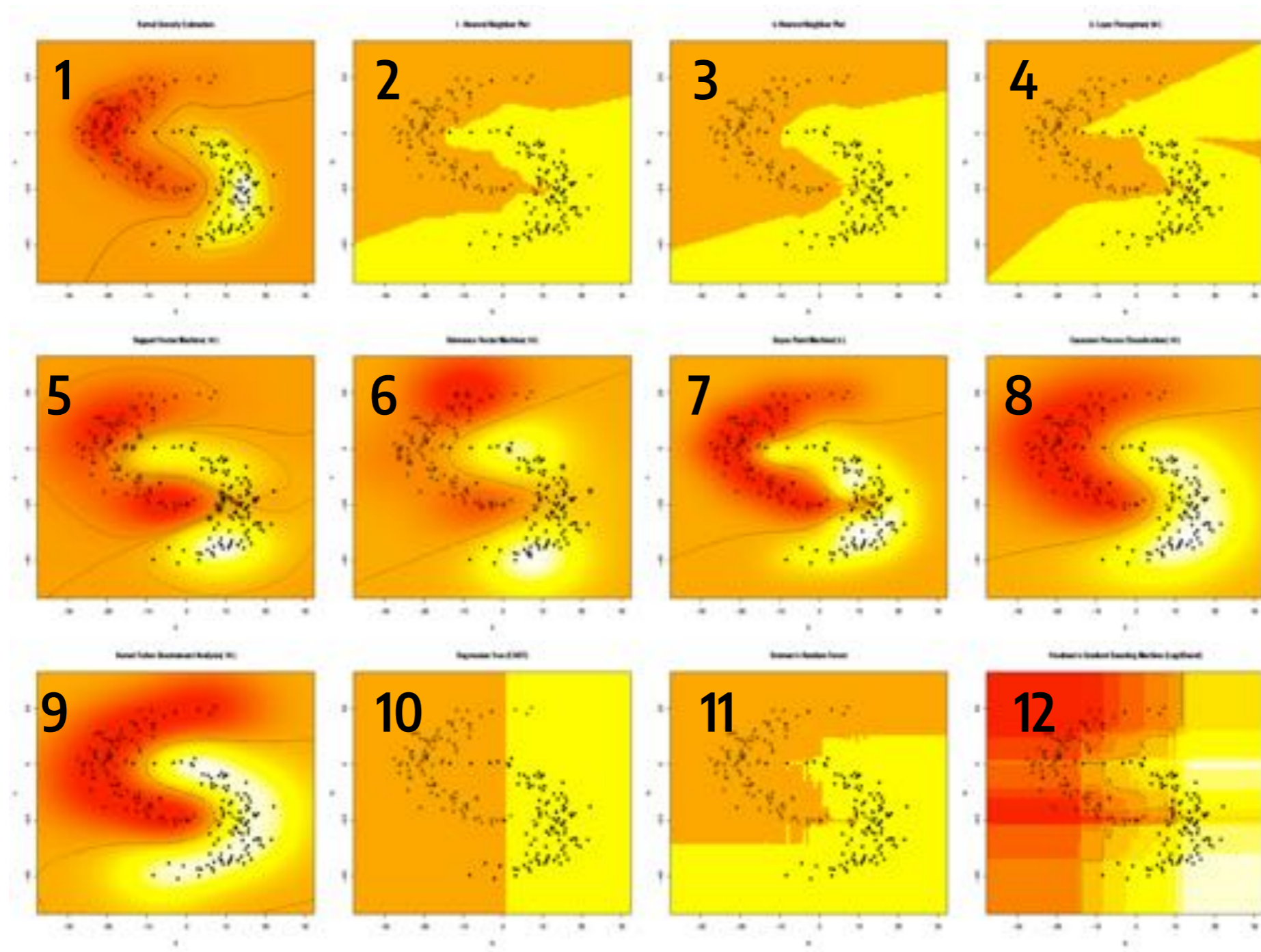
# Why ML looks so complicated?



(Generally high-dimensional rather than 1-dimensional)

- Many ways (ML models) exist for “curves (surfaces)”
- What models and input representations work best depends on the target problem
- We need to choose appropriate ML models for each given dataset and target problem

# Many ML models (Deep learning is one of them)



- |                              |                                |                                 |
|------------------------------|--------------------------------|---------------------------------|
| 1. Plugin Bayes Classifier   | 5. Support Vector Machine      | 9. Kernel Discriminant Analysis |
| 2. 1-Nearest Neighbor Method | 6. Relevance Vector Machine    | 10. Regression Tree (CART)      |
| 3. 5-Nearest Neighbor Method | 7. Bayes Point Machine         | 11. Random Forest               |
| 4. 3-Layer Neural Networks   | 8. Gaussian Process Classifier | 12. Gradient Boosting Machine   |

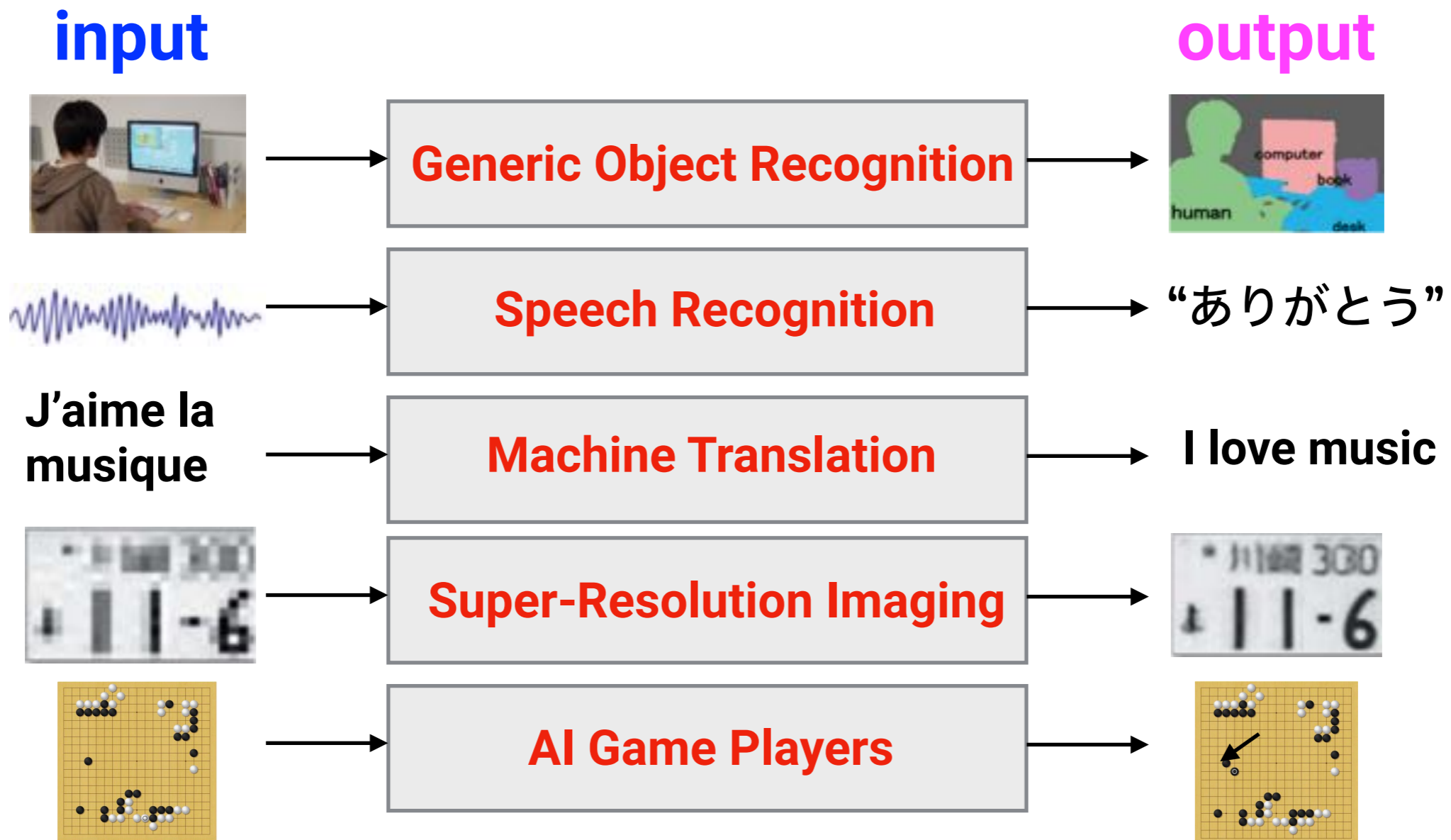
# Many ML models (and problems)

## Machine Learning Landscape

Supervised Learning	Unsupervised Learning	Others	
<b>Classification</b> <b>[Linear classification]</b> <ul style="list-style-type: none"><li>- Logistic / Softmax regression</li><li>- Linear discriminant analysis</li><li>- Naive Bayes classifiers</li><li>- Perceptron</li><li>- Linear Support Vector Machines (SVM)</li></ul> <b>[Nonlinear classification]</b> <ul style="list-style-type: none"><li>- k-nearest neighbor classifiers</li><li>- Decision trees (Classification trees)</li><li>- Polynomial classifiers / Factorization machines</li><li>- Tree ensemble classifiers<ul style="list-style-type: none"><li>- Random Forest classifiers</li><li>- Extra Trees classifiers</li><li>- Gradient Boosted Decision Trees (GBDT)</li></ul></li><li>- Kernel method classifiers<ul style="list-style-type: none"><li>- Support Vector Machines (SVM)</li></ul></li><li>- Gaussian process classifiers</li><li>- Neural network (Deep learning) classifiers<ul style="list-style-type: none"><li>- Multi-layer perceptrons (MLP)</li><li>- Convolutional networks (CNN)<ul style="list-style-type: none"><li>- VGG (OxfordNet)</li><li>- Inception (GoogLeNet)</li><li>- ResNet / ResNeXt</li><li>- DenseNet</li></ul></li><li>- Recurrent networks (RNN)</li></ul></li></ul>	<b>Regression</b> <b>[Linear regression]</b> <ul style="list-style-type: none"><li>- Least squares regression</li><li>- Principal component regression</li><li>- Partial Least Squares (PLS) regression</li><li>- Penalized linear regression<ul style="list-style-type: none"><li>- LASSO regression (L1-penalized)</li><li>- Ridge regression (L2-penalized)</li><li>- ElasticNet regression (L1 &amp; L2-penalized)</li></ul></li></ul> <b>[Nonlinear regression]</b> <ul style="list-style-type: none"><li>- k-nearest neighbor regressors</li><li>- Decision trees (Regression trees, Model trees)</li><li>- Polynomial regressors / Factorization machines</li><li>- Tree ensemble regressors<ul style="list-style-type: none"><li>- Random Forest regressors</li><li>- Extra Trees regressors</li><li>- Gradient Boosted Regression Trees (GBRT)</li></ul></li><li>- Kernel method regressors<ul style="list-style-type: none"><li>- Support Vector Regression (SVR)</li><li>- Kernel Ridge Regression</li></ul></li><li>- Gaussian process regressors</li><li>- Neural network (Deep learning) regressors<ul style="list-style-type: none"><li>- Multi-layer perceptrons (MLP)</li><li>- Convolutional networks (CNN)<ul style="list-style-type: none"><li>- VGG (OxfordNet)</li><li>- Inception (GoogLeNet)</li><li>- ResNet / ResNeXt</li><li>- DenseNet</li></ul></li><li>- Recurrent networks (RNN)</li></ul></li></ul>	<b>Clustering</b> <ul style="list-style-type: none"><li>- k-means</li><li>- Hierarchical clustering</li><li>- Gaussian mixtures</li><li>- Spectral methods</li><li>- DBSCAN</li></ul> <b>Decomposition</b> <ul style="list-style-type: none"><li>- Principal component analysis (PCA)</li><li>- Independent component analysis (ICA)</li><li>- Canonical correlation analysis (CCA)</li><li>- Nonnegative matrix factorization (NMF)</li><li>- Latent Dirichlet allocation (LDA)</li></ul> <b>Manifold learning</b> <ul style="list-style-type: none"><li>- Multidimensional scaling (MDS)</li><li>- Self-organizing maps (SOM)</li><li>- Isomap</li><li>- Locally linear embedding (LLE)</li><li>- Spectral embedding (Laplacian eigenmaps)</li><li>- t-distributed Stochastic Neighbor Embedding (t-SNE)</li><li>- Autoencoders</li></ul> <b>Density estimation</b>	<b>Semi supervised learning</b> <b>Ranking</b> <b>Transfer learning</b> <b>K-shot learning</b> <b>Domain adaptation</b> <b>Multitask learning</b> <b>Reinforcement learning</b> <b>Active learning</b> <b>Model-based optimization</b> <b>Time series/Sequence models</b> <b>Probabilistic inference (Bayesian, Generative, Graphical)</b> <b>Causal inference</b> <b>Online/Incremental learning</b> <b>Anomaly/Outlier detection</b> <b>Ensemble learning</b> <b>Relational/Network learning</b> <b>Representation learning</b> <b>Structured prediction</b> <b>Meta Learning</b> :

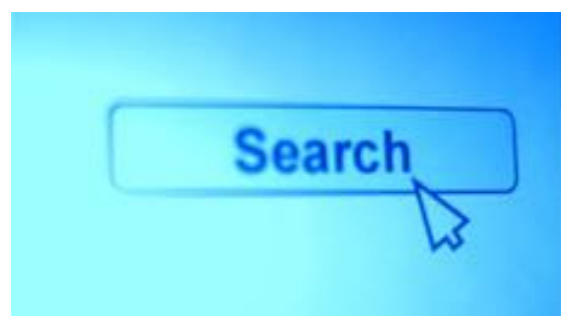
# (Supervised) machine learning in other words

Machine learning is a systematic way to find **a high-dimensional mapping** from **input** to **output** just by giving a lot of **input-output** "examples".



# This lazy idea really works in diverse applications

**Search Engine**



**Translation**



**Self-driving**



**Medicine**



**Advertising**



**Transportation**



**Weather**



**Security**



**Smart devices, IoT, e-Commerce, Manufacturing, Agriculture, Disaster Prevention, Finance, Education, Employment, Matchmaking, and, of course, Science.**

Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
  - "The quality of your inputs decide the quality of your output"
  - Training / validation / test data
  - Tuning hyperparameters
  - Identification and design of input variables (or "descriptors")
  - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions



# Machine learning for science

The current interests and targets of sciences becomes more complex than ever, and simply put, we need a new way to break through this trend.

Nature, 559  
pp. 547–555 (2018)

Science, 361  
pp. 360-365 (2018)

Science, 293  
pp. 2051-2055 (2001)

## REVIEW

<https://doi.org/10.1038/s41586-018-0337-2>

### Machine learning for molecular and materials science

Keith T. Butler<sup>1</sup>, Daniel W. Davies<sup>2</sup>, Hugh Cartwright<sup>3</sup>, Olexandr Isayev<sup>4\*</sup> & Aron Walsh<sup>5,6\*</sup>

Here we summarize recent progress in machine learning for the chemical sciences. We outline machine-learning techniques that are suitable for addressing research questions in this domain, as well as future directions for the field. We envisage a future in which the design, synthesis, characterization and application of molecules and materials is accelerated by artificial intelligence.

The Schrödinger equation provides a powerful structure–property relationship for molecules and materials. For a given spatial arrangement of chemical elements, the distribution of electrons and a wide range of physical responses can be described. The development of quantum mechanics provided a rigorous theoretical foundation for the chemical bond. In 1929, Paul Dirac famously proclaimed that the underlying physical laws for the whole of chemistry are “completely known”<sup>1</sup>. John Pople, realizing the importance of rapidly developing computer technologies, created a program—Gaussian 70—that could perform ab initio calculations: predicting the behaviour, for molecules of modest size, purely from the fundamental laws of physics<sup>2</sup>. In the 1960s, the Quantum Chemistry Program Exchange brought quantum chemistry to the masses in the form of useful practical tools<sup>3</sup>. Suddenly, experimentalists with little or no theoretical training could perform quantum calculations too. Using modern algorithms and supercomputers, systems containing thousands of interacting ions and electrons can now be described using approximations to the physical laws that govern the world on the atomic scale<sup>4,5</sup>.

The field of computational chemistry has become increasingly predictive in the twenty-first century, with activity in applications as wide ranging as catalyst development for greenhouse gas conversion, materials discovery for energy harvesting and storage, and computer-assisted drug design<sup>6</sup>. The modern chemical-simulation toolkit allows the properties of a compound to be anticipated (with reasonable accuracy) before it has been made in the laboratory. High-throughput computational screening has become routine, giving scientists the ability to calculate the properties

of thousands of computational functional theory (DFT) structures and behavior, using extensive databases that hypothetical systems, in molecules and metal alloys.

The emergence of computational science and engineering, and the “fourth industrial revolution” in the chemical domain, artificial intelligence that learning. At the heart of machine learning is a growing

generating, testing and refining scientific models. Such techniques are suitable for addressing complex problems that involve massive combinatorial spaces or nonlinear processes, which conventional procedures either cannot solve or can tackle only at great computational cost.

As the machinery for artificial intelligence and machine learning matures, important advances are being made not only by those in mainstream artificial-intelligence research, but also by experts in other fields (domain experts) who adopt these approaches for their own purposes. As we detail in Box 1, the resources and tools that facilitate the application of machine-learning techniques mean that the barrier to entry is lower than ever.

In the rest of this Review, we discuss progress in the application of machine learning to address challenges in molecular and materials research. We review the basics of machine-learning approaches, identifying areas in which existing methods have the potential to accelerate research and consider the developments that are required to enable more wide-ranging impacts.

**Nuts and bolts of machine learning**  
With machine learning, given enough data and a rule-discovery algorithm, a computer has the ability to determine all known physical laws (and potentially those that are currently unknown) without human input. In traditional computational approaches, the computer is little more than a calculator, employing a hard-coded algorithm provided by a human expert. By contrast, machine-learning approaches learn the rules that underlie a dataset by assessing a portion of that data

## SPECIAL SECTION FRONTIERS IN COMPUTATION

### REVIEW

### Inverse molecular design using machine learning: Generative models for matter engineering

Benjamin Sanchez-Lengeling<sup>1</sup> and Alán Aspuru-Guzik<sup>2,3,4\*</sup>

The discovery of new materials can bring enormous societal and technological progress. In this context, exploring completely the large space of potential materials is computationally intractable. Here, we review methods for achieving inverse design, which aims to discover tailored materials from the starting point of a particular desired functionality. Recent advances from the rapidly growing field of artificial intelligence, mostly from the subfield of machine learning, have resulted in a fertile exchange of ideas, where approaches to inverse molecular design are being proposed and employed at a rapid pace. Among these, deep generative models have been applied to numerous classes of materials: rational design of prospective drugs, synthetic routes to organic compounds, and optimization of photovoltaics and redox flow batteries, as well as a variety of other solid-state materials.

Many of the challenges of the 21st century (1), from personalized health care to energy production and storage, share a common theme: materials are part of the solution (2). In some cases, the solutions to these challenges are fundamentally limited by the physics and chemistry of a material, such as the relationship of a materials bandgap to the thermodynamic limits for the generation of solar energy (3).

Several important materials discoveries arose by chance or through a process of trial and error. For example, vulcanized rubber was prepared in the 19th century from random mixtures of compounds, based on the observation that heating with additives such as sulfur improved the rubber's durability. At the molecular level, individual polymer chains cross-linked, forming bridges that enhanced the macroscopic mechanical properties (4). Other notable examples in this vein include Teflon, anesthesia, Vaseline, Penicillin's mauve, and penicillin. Furthermore, these materials come from common chemical

mapped 166.4 billion molecules that contain at most 17 heavy atoms. For pharmacologically relevant small molecules, the number of structures is estimated to be on the order of  $10^{60}$  (5). Adding consideration of the hierarchy of scale from subnanometer to microscopic and mesoscopic further complicates exploration of chemical space in its entirety (10). Therefore, any global strategy for covering this space might seem impossible.

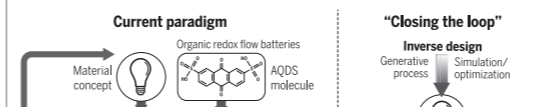
Simulation offers one way of probing this space without experimentation. The physics and chemistry of these molecules are governed by quantum mechanics, which can be solved via the Schrödinger equation to arrive at their ex-

act properties. In practice, approximations are used to lower computational time at the cost of accuracy.

Although theory enjoys enormous progress, now routinely modeling molecules, clusters, and perfect as well as defect-laden periodic solids, the size of chemical space is still overwhelming, and smart navigation is required. For this purpose, machine learning (ML), deep learning (DL), and artificial intelligence (AI) have a potential role to play because their computational strategies automatically improve through experience (11). In the context of materials, ML techniques are often used for property prediction, seeking to learn a function that maps a molecular material to the property of choice. Deep generative models are a special class of DL methods that seek to model the underlying probability distribution of both structure and property and relate them in a nonlinear way. By exploiting patterns in massive datasets, these models can distill average and salient features that characterize molecules (12, 13).

Inverse design is a component of a more complex materials discovery process. The time scale for deployment of new technologies, from discovery in a laboratory to a commercial product, historically, is 15 to 20 years (14). The process (Fig. 1) conventionally involves the following steps: (i) generate a new or improved material concept and simulate its potential suitability; (ii) synthesize the material; (iii) incorporate the material into a device or system; and (iv) characterize and measure the desired properties. This cycle generates feedback to repeat, improve, and refine future cycles of discovery. Each step can take up to several years.

In the era of matter engineering, scientists seek to accelerate these cycles, reducing the



**Table 1.** Parallels between genome sequencing and genetic network discovery.

Genome sequencing	Genome semantics
Physical maps	Graphical model
Contigs	Low-level functional models
Contig reassembly	Module assembly
Finished genome sequence	Comprehensive model

DNA to be sequences into distinct pieces, parcel out the detailed work of sequencing, and then reassemble these independent efforts at the end. It is not quite so simple in the world of genome semantics.

Despite the differences between genome sequencing and genetic network discovery, there are clear parallels that are illustrated in Table 1. In genome sequencing, a physical map is useful to provide scaffolding for assembling the finished sequence. In the case of a genetic regula-

## COMPUTERS AND SCIENCE

tory network, a graphical model can play the same role. A graphical model can represent a high-level view of interconnectivity and help isolate modules that can be studied independently. Like contigs in a genomic sequencing project, low-level functional models can explore the detailed behavior of a module of genes in a manner that is consistent with the higher level graphical model of the system. With standardized nomenclature and compatible modeling techniques, independent functional models can be assembled into a complete model of the cell under study.

To enable this process, there will need to be standardized forms for model representation. At present, there are many different modeling technologies in use, and although models can be easily placed into a database, they are not useful out of the context of their specific modeling package. The need for a standardized way of communicating computational descriptions of biological systems extends to the literature. Entire conferences have been established to explore ways of mining the biology literature to extract se-

semantic information in computational form. Going forward, as a community we need to come to consensus on how to represent what we know about biology in computational form as well as in words. The key to postgenomic biology will be the computational assembly of our collective knowledge into a cohesive picture of cellular and organism function. With such a comprehensive model, we will be able to explore new types of conservation between organisms and make great strides toward new therapeutics that function on well-characterized pathways.

### References

1. S. K. Kim et al., *Science* **293**, 2087 (2001).
2. A. Hartemink et al., paper presented at the Pacific Symposium on Biocomputing 2000, Oahu, Hawaii, 4 to 9 January 2000.
3. D. Pe'er et al., paper presented at the 9th Conference on Intelligent Systems in Molecular Biology (ISM'99), Copenhagen, Denmark, 21 to 25 July 2001.
4. H. McAdams, A. Arkin, *Proc. Natl. Acad. Sci. U.S.A.* **94**, 814 (1997).
5. A. J. Hartemink, thesis, Massachusetts Institute of Technology, Cambridge (2001).

## VIEWPOINT

### Machine Learning for Science: State of the Art and Future Prospects

Eric Mjølness\* and Dennis DeCoste

Recent advances in machine learning methods, along with successful applications across a wide variety of fields such as planetary science and bioinformatics, promise powerful new tools for practicing scientists. This viewpoint highlights some useful characteristics of modern machine learning methods and their relevance to scientific applications. We conclude with some speculations on near-term progress and promising directions.

Machine learning (ML) (1) is the study of computer algorithms capable of learning to improve their performance of a task on the basis of their own previous experience. The field is

correlate surprisingly well with subsequent gene expression analysis (3). Postgenomic biology prominently features large-scale gene expression data analyzed by clustering methods

creating hypotheses, testing by decisive experiment or observation, and iteratively building up comprehensive testable models or theories is shared across disciplines. For each stage of this abstracted scientific process, there are relevant developments in ML, statistical inference, and pattern recognition that will lead to semiautomatic support tools of unknown but potentially broad applicability.

Increasingly, the early elements of scientific method—observation and hypothesis generation, high data acquisition for objective analysis by human perception—the situation in experiments for decades. There is significant need, including Hough foundational in pattern analysis (8) used in neutrino Microscope imagery in pathology, and other processing specialties, from Earth-observing newly operational Terra MODIS (imaging

Science is changing, the tools of science are changing. And that requires different approaches. — Erich Bloch, 1925-2016

<sup>1</sup>ISIS Facility, Rutherford Appleton Laboratory, Didcot, UK. \*Corresponding author. Email: alexandr@olexandrisayev.com; a.walsh@imperial.ac.uk

\*Corresponding author. Email: aspuru@utoronto.ca

(93). AI/ML, embedded systems, and robotics (87) into an integrated ecosystem.

# Empirical optimization or "Edisonian empiricism"



**Problem: time and cost when we use this for all possible candidates**



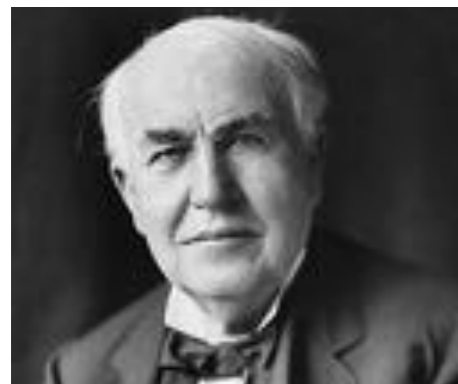
**feedback to next plans**

**available data and findings**

**Generate hypotheses**

- Experiments
- Simulations

**Check and validate results**



**Thomas Edison**

- Genius is 1% inspiration and 99% perspiration.
- There is no substitute for hard work.
- I have not failed. I've just found 10,000 ways that won't work.

:



**Experience + Intuition = Incredible hardwork + Luck**



# Empirical optimization or "Edisonian empiricism"



**Problem: time and cost when we use this for all possible candidates**



**feedback to next plans**

**available data and findings**

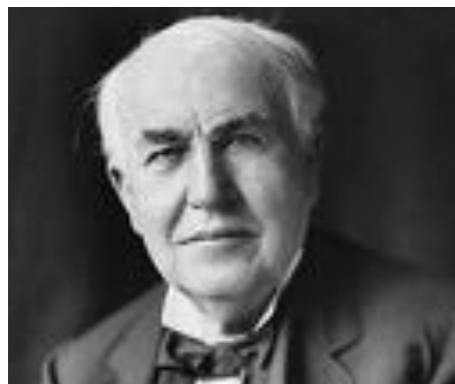
Ge  
hy

But, wait. Is this still valid now in 2020?

Everything looks complicated than Edison thought, and how "incredibly hard" work do I need...?



ts



**Thomas Edison**

that work work.

:

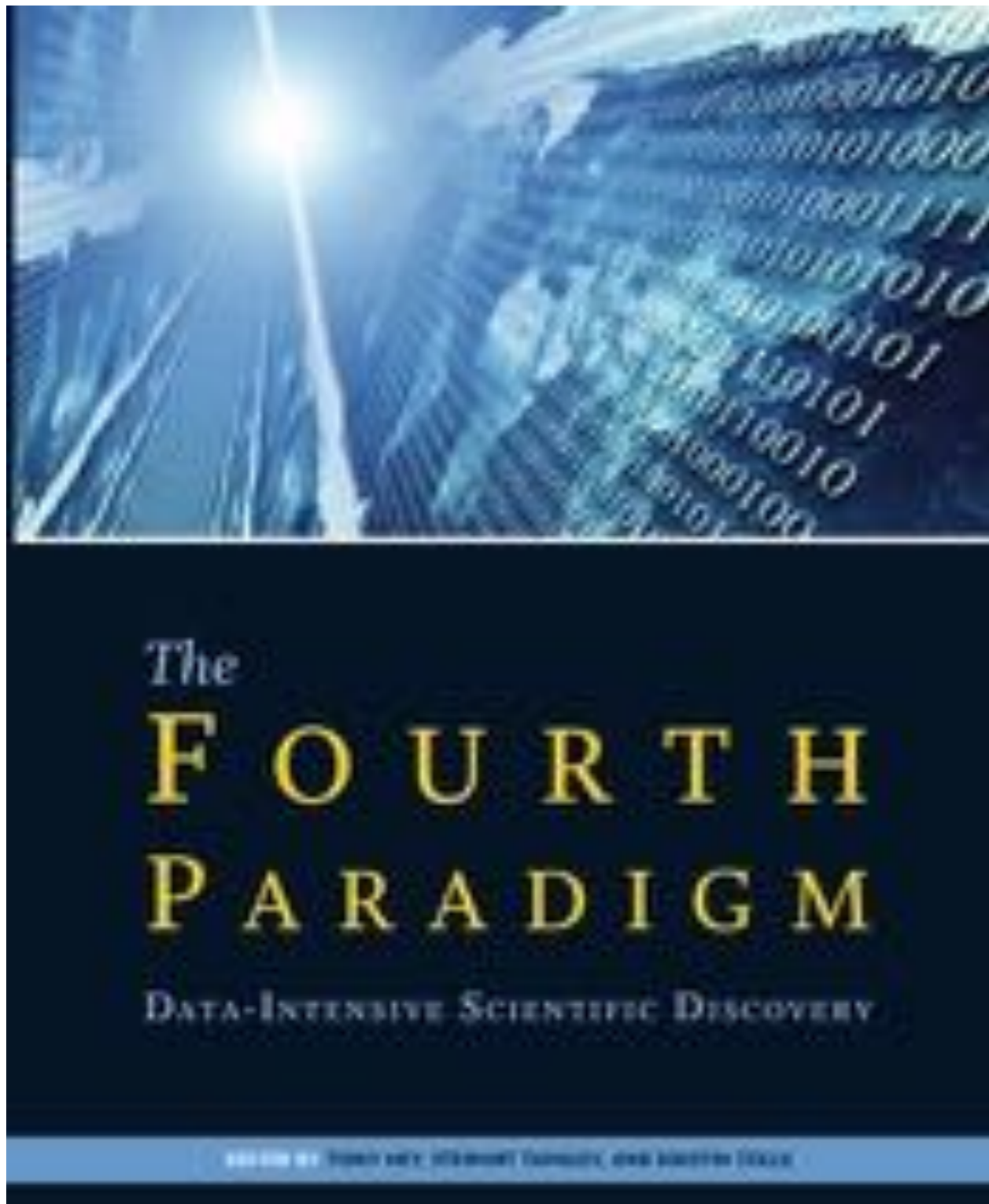


**Experience + Intuition = Incredible hardwork + Luck**



<http://www.fourthparadigm.org/>

by [Tony Hey](#), Stewart Tansley, Kristin Tolle



**Tony Hey**

In *The Fourth Paradigm: Data-Intensive Scientific Discovery*, the collection of essays expands on **the vision of pioneering computer scientist Jim Gray** for a new, fourth paradigm of discovery based on **data-intensive science** and offers insights into how it can be fully realized.

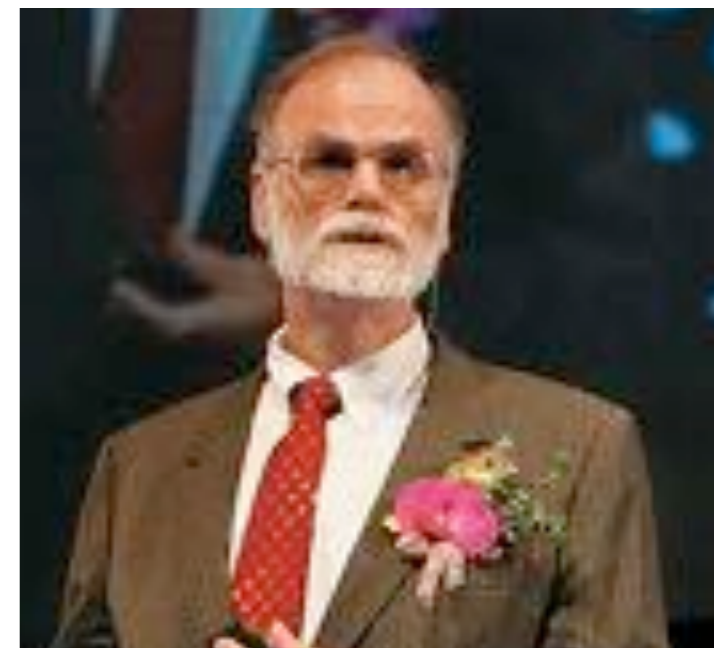
# Jim Gray on "eScience"

<http://research.microsoft.com/en-us/um/people/gray/JimGrayTalks.htm>

["eScience" Talk at NRC-CSTB meeting](#)  
Mountain View CA, 11 January 2007.

His 'last' talk before the [disappearance](#)

On January 28, 2007 he failed to return from a short solo trip to the Farallon Islands near San Francisco to scatter his mother's ashes.



Jim Gray

NRC = National Research Council

<http://sites.nationalacademies.org/NRC/index.htm>;

CSTB = Computer Science and Telecom- munications Board

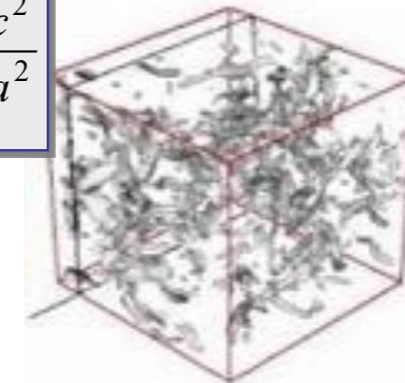
<http://sites.nationalacademies.org/cstb/index.htm>.

# Science Paradigms

- Thousand years ago:  
science was **empirical**  
describing natural phenomena
- Last few hundred years:  
**theoretical** branch  
using models, generalizations
- Last few decades:  
a **computational** branch  
simulating complex phenomena
- Today:  
**data exploration (eScience)**  
unify theory, experiment, and simulation
  - Data captured by instruments  
Or generated by simulator
  - Processed by software
  - Information/Knowledge stored in computer
  - Scientist analyzes database / files  
using data management and statistics

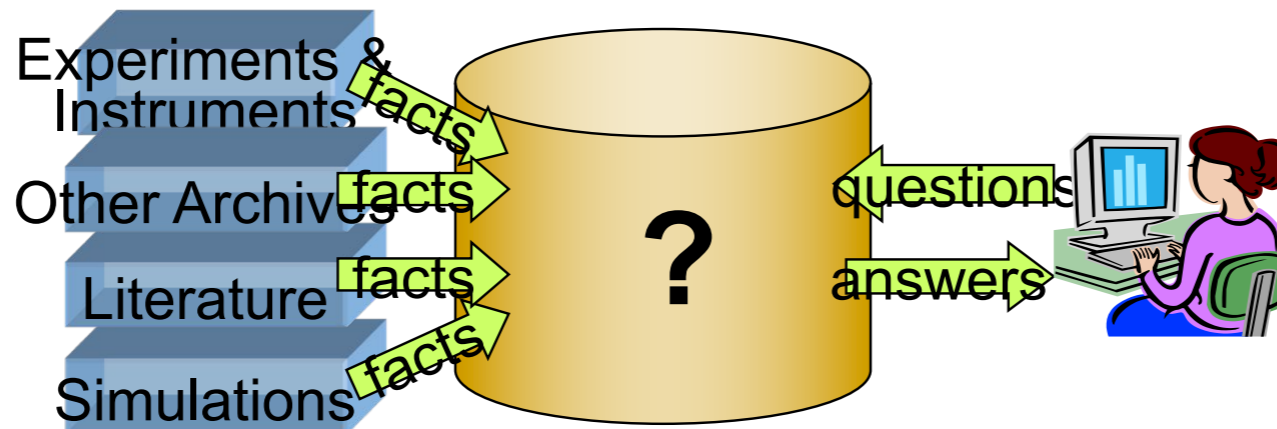


$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi G\rho}{3} - K\frac{c^2}{a^2}$$



# X-Info

- The evolution of X-Info and Comp-X  
for each discipline X
- How to codify and represent our knowledge



## The Generic Problems

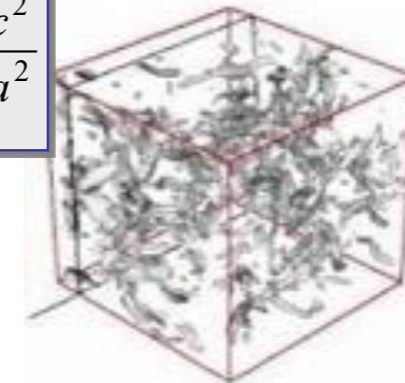
- Data ingest
- Managing a petabyte
- Common schema
- How to organize it
- How to *reorganize* it
- How to share with others
- Query and Vis tools
- Building and executing models
- Integrating data and Literature
- Documenting experiments
- Curation and long-term preservation

# Science Paradigms

- Thousand years ago:  
science was **empirical**  
describing natural phenomena
- Last few hundred years:  
**theoretical** branch  
using models, generalizations
- Last few decades:  
a **computational** branch  
simulating complex phenomena
- Today:  
**data exploration (eScience)**  
unify theory, experiment, and simulation
  - Data captured by instruments  
Or generated by simulator
  - Processed by software
  - Information/Knowledge stored in computer
  - Scientist analyzes database / files  
using data management and statistics



$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi G\rho}{3} - K\frac{c^2}{a^2}$$



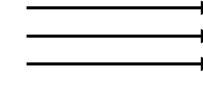
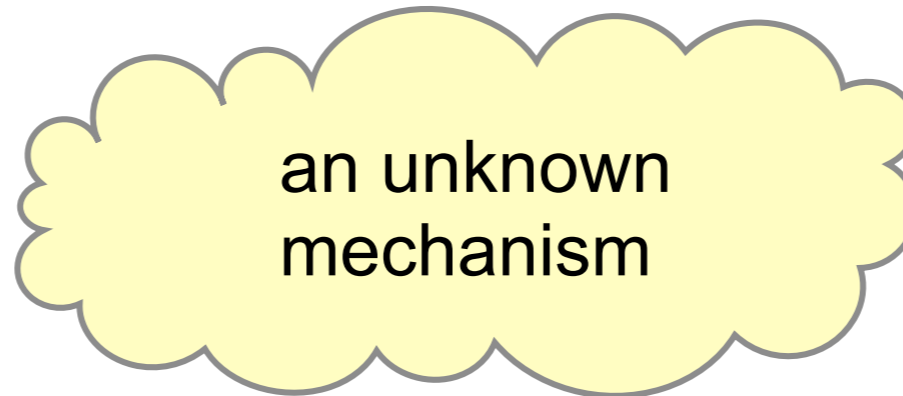
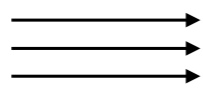


- Today:
  - data exploration (eScience)**
  - unify theory, experiment, and simulation
  - Data captured by instruments
  - Or generated by simulator
  - Processed by software
  - Information/Knowledge stored in computer
  - Scientist analyzes database / files using data management and statistics

↳ **This would be more strongly complemented by Machine learning or AI, or data sciences.**

# The interest of science

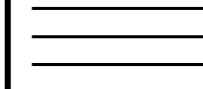
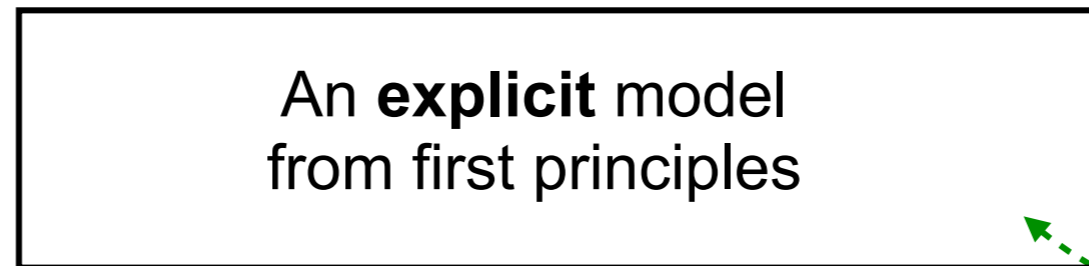
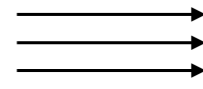
Related factors  
and their states



observations  
(data)

## • Theory-driven (rational, deductive) approach

Related factors  
and their states



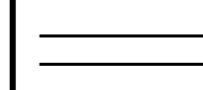
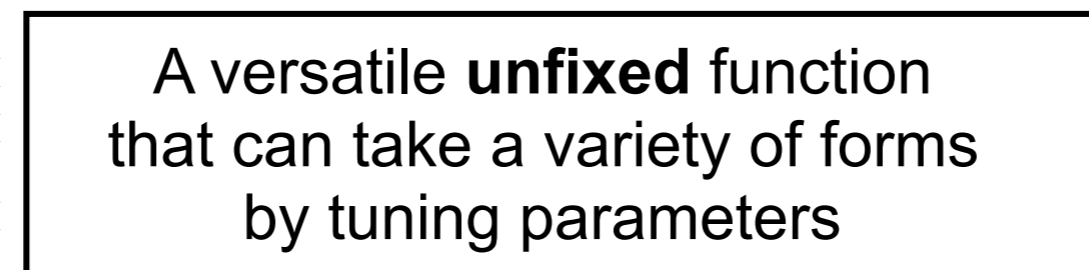
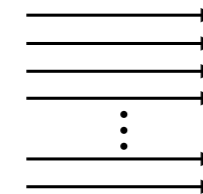
observations  
(data)

Ideally, the smallest number  
of dominant factors

Whether or not theory is correct can be validated

## • Data-driven (empirical, inductive) approach

Related factors  
and their states



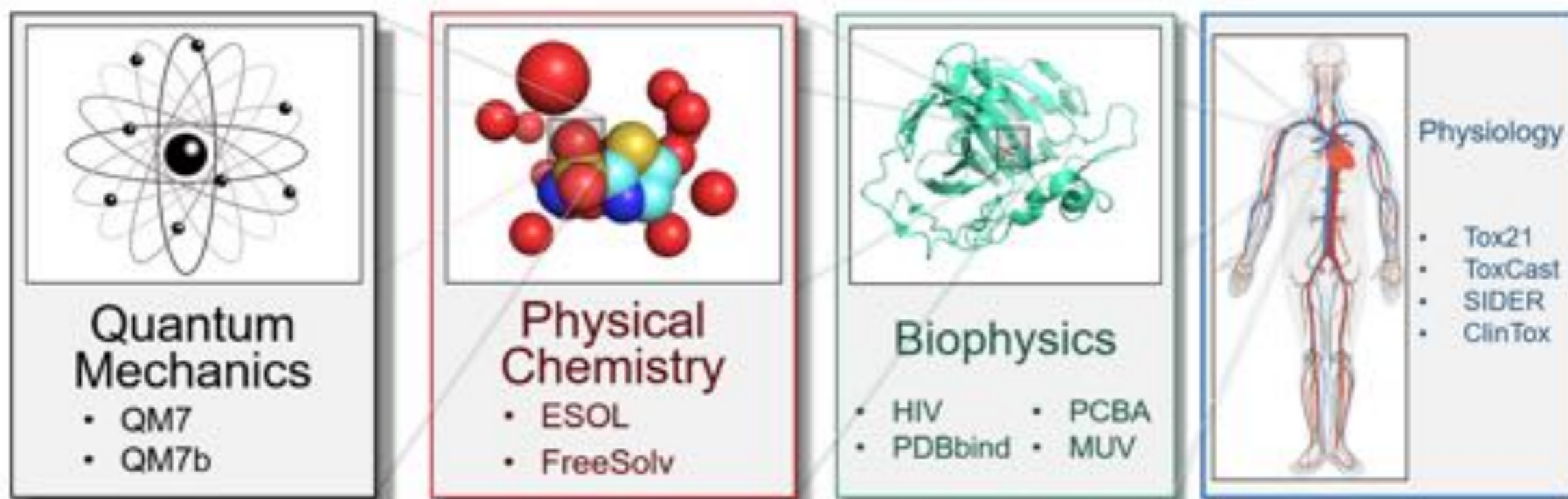
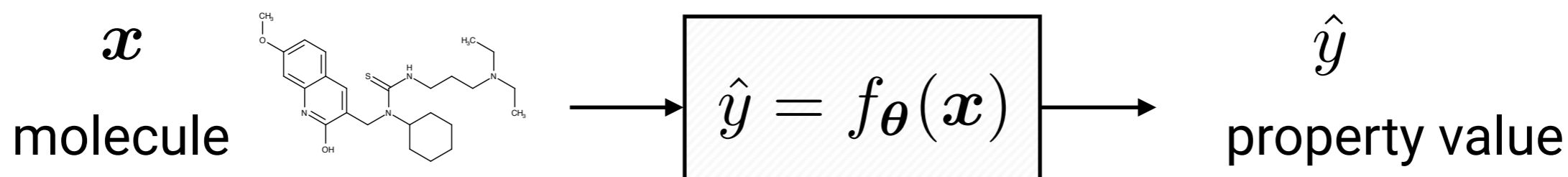
observations  
(data)

All potentially related factors  
(the number can be very large  
as long as they are sensible)

Function is best fitted to data by tuning parameters

# Predictive modeling by machine learning

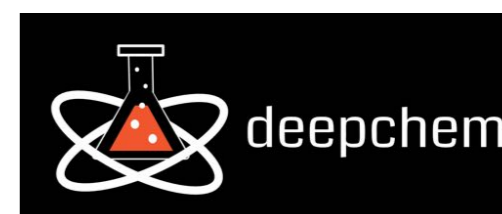
## Quantitative structure–activity/property relationship (QSAR/QSPR)



*MoleculeNet: A Benchmark for Molecular Machine Learning*

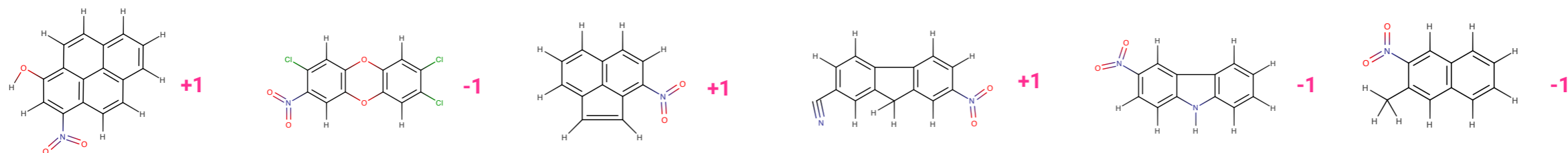
<https://arxiv.org/abs/1703.00564>

<https://github.com/deepchem/deepchem> (<https://deepchem.io/>)

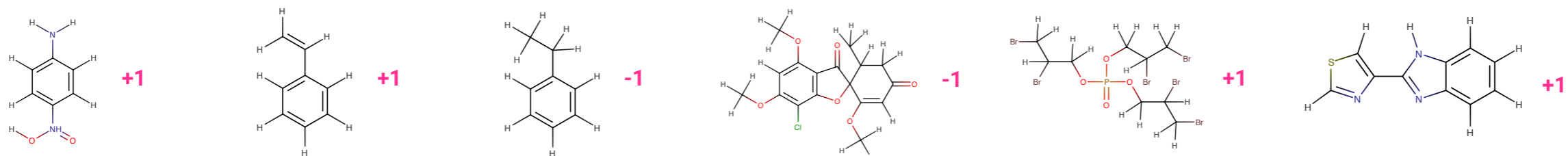


# Many levels of interest

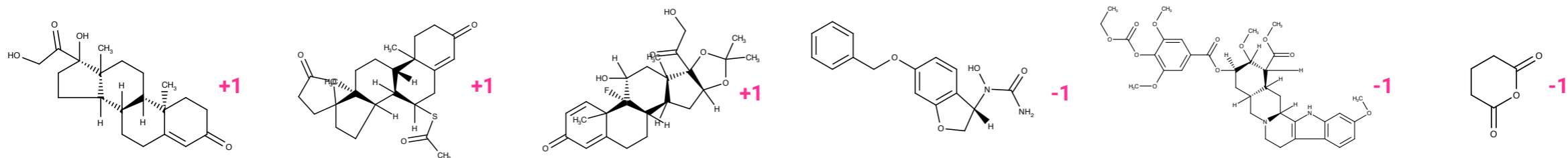
- **Mutagenic potency**



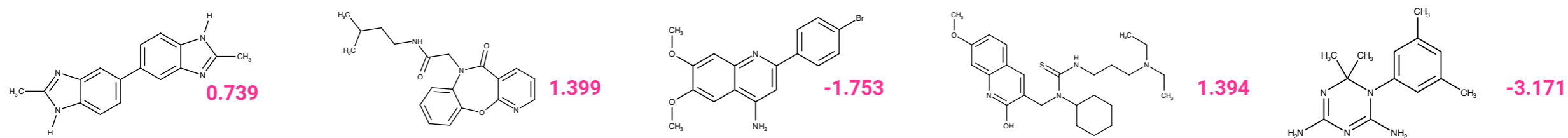
- **Carcinogenic potency**



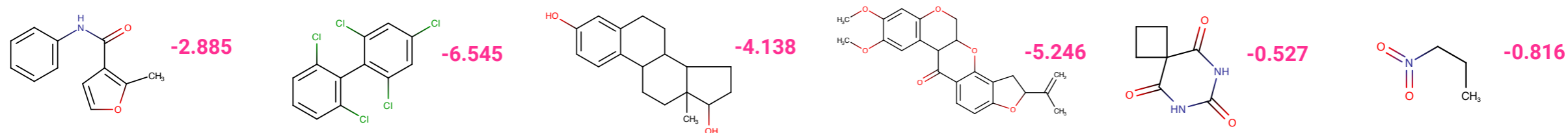
- **Endocrine disruption**



- **Growth inhibition**



- **Aqueous solubility**

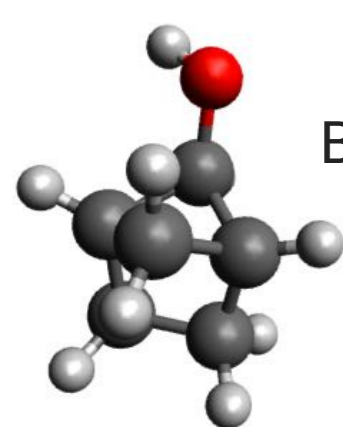


# Data-driven approximation to quantum chem

Quantum chemistry structures and properties of 134 kilo molecules, *Scientific Data* 1, 140022 (2014)

<http://www.nature.com/articles/sdata201422>

<http://quantum-machine.org/datasets/>



DFT

B3LYP/6-31G(2df,p)



$U_0, U, H, G$   
 $\omega_1, \text{ZPVE}$   
 $\epsilon_{\text{HOMO}}, \epsilon_{\text{LUMO}}, \Delta\epsilon$   
 $\langle R^2 \rangle, \mu, \alpha$

Structure in  
the ground state

15 properties in  
the ground state

1  HOMO: -0.3877 LUMO: 0.1171 Dipole_moment: 0.0	2  HOMO: -0.257 LUMO: 0.0829 Dipole_moment: 1.6256	3  HOMO: -0.2928 LUMO: 0.0687 Dipole_moment: 1.8511	4  HOMO: -0.2845 LUMO: 0.0506 Dipole_moment: 0.0	5  HOMO: -0.3604 LUMO: 0.0191 Dipole_moment: 2.8937
6  HOMO: -0.267 LUMO: -0.0406 Dipole_moment: 2.1889	7  HOMO: -0.3385 LUMO: 0.1041 Dipole_moment: 0.0	8  HOMO: -0.2653 LUMO: 0.0784 Dipole_moment: 1.5258	9  HOMO: -0.2609 LUMO: 0.0613 Dipole_moment: 0.7156	10  HOMO: -0.3264 LUMO: 0.0376 Dipole_moment: 3.8266
11  HOMO: -0.254 LUMO: -0.0198 Dipole_moment: 2.5682	12  HOMO: -0.2543 LUMO: 0.0302 Dipole_moment: 3.7286	13  HOMO: -0.323 LUMO: 0.0949 Dipole_moment: 0.0597	14  HOMO: -0.2619 LUMO: 0.0798 Dipole_moment: 1.4131	15  HOMO: -0.2525 LUMO: 0.091 Dipole_moment: 1.1502
16  HOMO: -0.2888 LUMO: 0.1042 Dipole_moment: 5.0E-4	17  HOMO: -0.2682 LUMO: 0.1042 Dipole_moment: 1.7675	18  HOMO: -0.2431 LUMO: -0.0087 Dipole_moment: 2.7362	19  HOMO: -0.2436 LUMO: 0.0347 Dipole_moment: 3.6367	20  HOMO: -0.2495 LUMO: 0.0556 Dipole_moment: 3.4869
21  HOMO: -0.3167 LUMO: 0.0843 Dipole_moment: 0.0897	22  HOMO: -0.2612 LUMO: 0.074 Dipole_moment: 1.4259	23  HOMO: -0.2599 LUMO: -0.0214 Dipole_moment: 0.0	24  HOMO: -0.3102 LUMO: -0.0543 Dipole_moment: 3.792	25  HOMO: -0.3696 LUMO: -0.0926 Dipole_moment: 0.0023



JCTC

Journal of Chemical Theory and Computation

Cite This: *J. Chem. Theory Comput.* 2017, 13, 5255-5264

Article

[pubs.acs.org/JCTC](https://pubs.acs.org/JCTC)

## Prediction Errors of Molecular Machine Learning Models Lower than Hybrid DFT Error

Felix A. Faber,<sup>†</sup> Luke Hutchison,<sup>‡</sup> Bing Huang,<sup>†</sup> Justin Gilmer,<sup>‡</sup> Samuel S. Schoenholz,<sup>‡</sup> George E. Dahl,<sup>‡</sup> Oriol Vinyals,<sup>‡</sup> Steven Kearnes,<sup>‡</sup> Patrick F. Riley,<sup>‡</sup> and O. Anatole von Lilienfeld<sup>\*,†,‡</sup>

# PERSPECTIVES

nature reviews chemistry

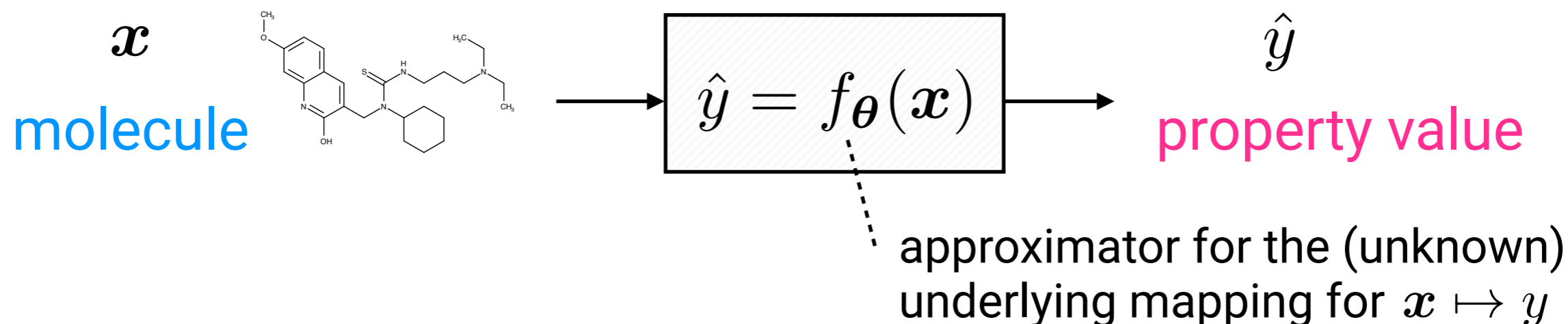
## Exploring chemical compound space with quantum-based machine learning

*Nat Rev Chem* **4**, 347–358 (2020)

O. Anatole von Lilienfeld, Klaus-Robert Müller and Alexandre Tkatchenko 

**Abstract** | Rational design of compounds with specific properties requires understanding and fast evaluation of molecular properties throughout chemical compound space — the huge set of all potentially stable molecules. Recent advances in combining quantum-mechanical calculations with machine learning provide powerful tools for exploring wide swathes of chemical compound space. We present our perspective on this exciting and quickly developing field by discussing key advances in the development and applications of quantum-mechanics-based machine-learning methods to diverse compounds and properties, and outlining the challenges ahead. We argue that significant progress in the exploration and understanding of chemical compound space can be made through a systematic combination of rigorous physical theories, comprehensive synthetic data sets of microscopic and macroscopic properties, and modern machine-learning methods that account for physical and chemical knowledge.

# “Molecular Machine Learning”



Given  $n$  input-output instances (as the training data)

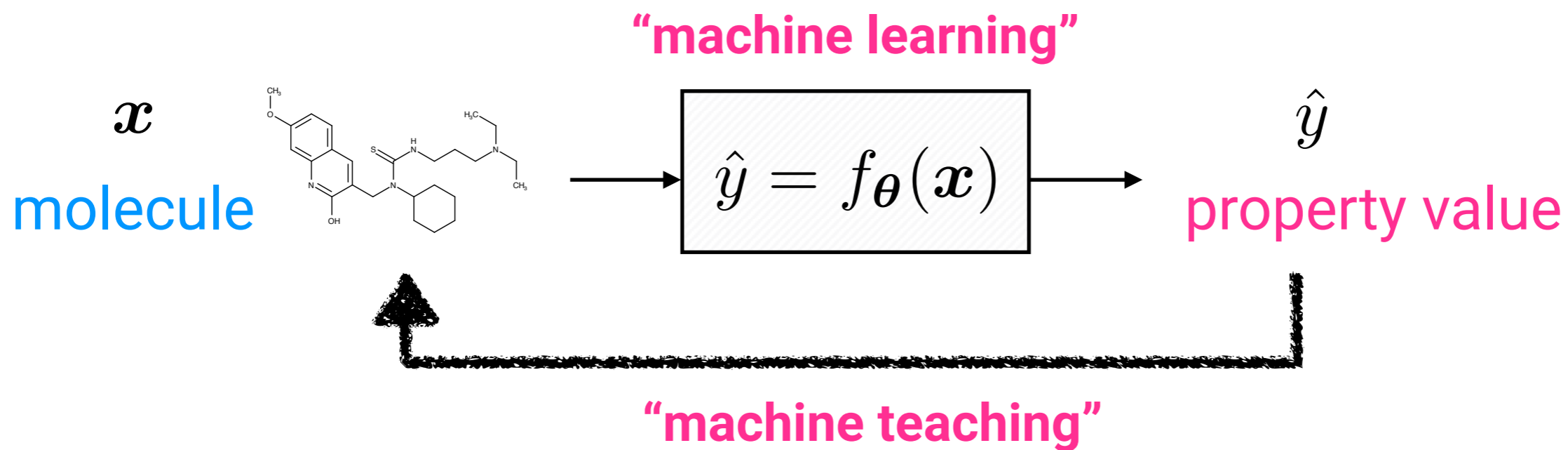
$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

Fit the model  $f_{\theta}$  by tuning  $\theta$  as

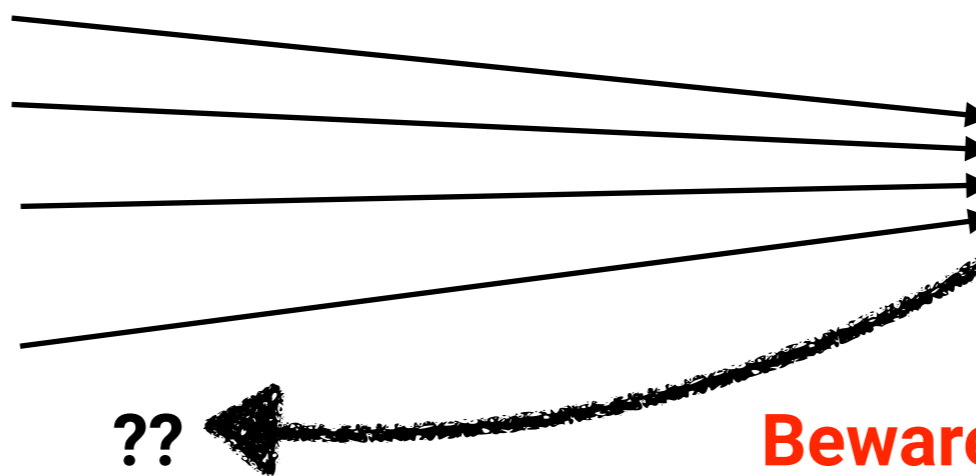
$$\min_{\theta} \sum_{i=1}^n \text{error}(y_i, \hat{y}_i) \quad \text{where} \quad \hat{y}_i = f_{\theta}(x_i)$$

Note: the error measure (called "loss function" in ML) depends on problems

# Inverse design?



**Bayesian optimization / Black-box optimization / Sequential design of experiments / Model-based optimization**



**“television”**

??

**Beware: Inverse is not unique**



Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
  - "The quality of your inputs decide the quality of your output"
  - Training / validation / test data
  - Tuning hyperparameters
  - Identification and design of input variables (or "descriptors")
  - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

# Now is the good timing to start machine learning!

**Let's give it a try anyway!**

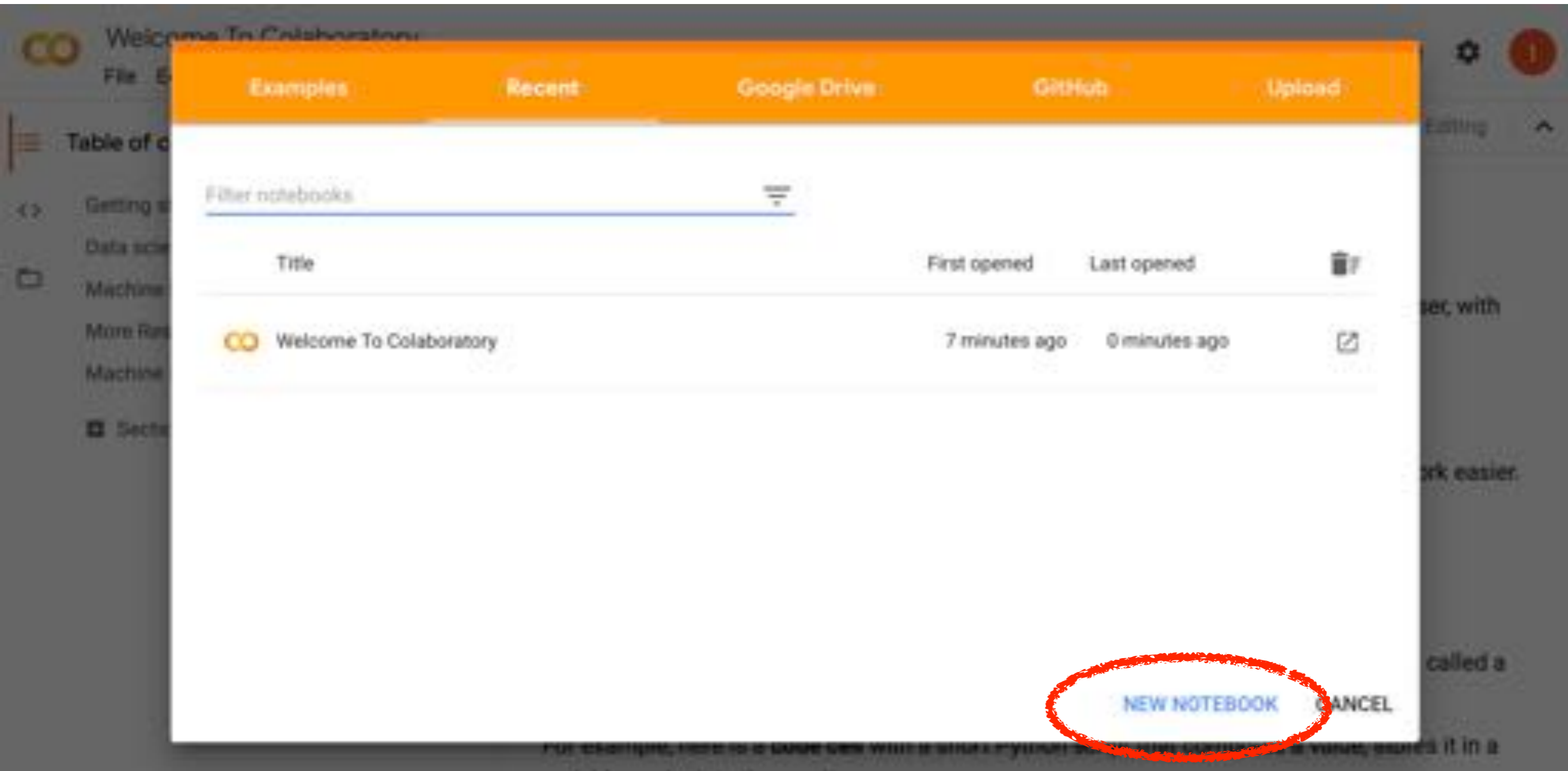
**All you need for now is a **web browser** only.  
(on your PC or smartphones)**

<https://colab.research.google.com/>



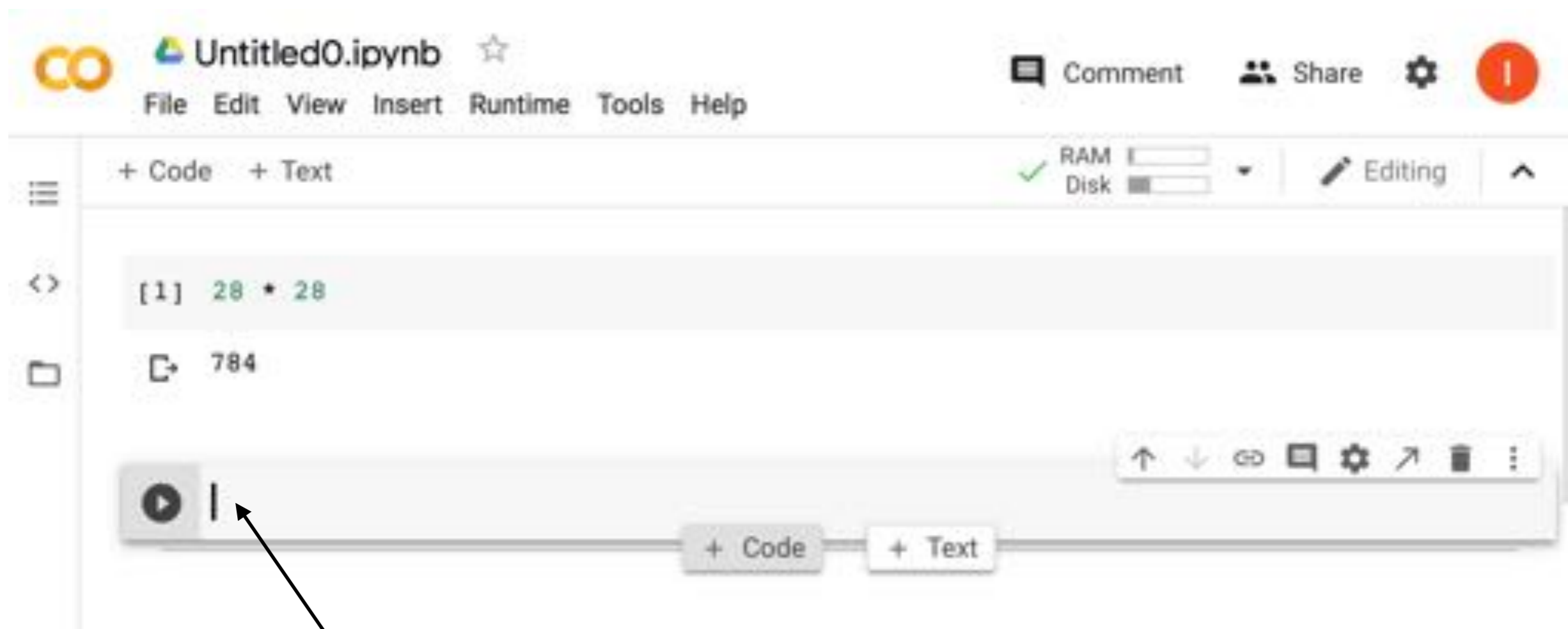
# Google Colab (Google Colaboratory)

<https://colab.research.google.com/>



# Google Colab (Google Colaboratory)

<https://colab.research.google.com/>



You can input a **Python** code here and run it by clicking the  button (or by Shift+Enter)

# Let's reproduce a paper-level result!

RSC Advances



PAPER

[View Article Online](#)

[View Journal](#) | [View Issue](#)



Cite this: *RSC Adv.*, 2016, 6, 52587

## Machine-learning prediction of the d-band center for metals and bimetals

Ichigaku Takigawa,<sup>\*ab</sup> Ken-ichi Shimizu,<sup>cd</sup> Koji Tsuda<sup>efg</sup> and Satoru Takakusagi<sup>c</sup>

The d-band center for metals has been widely used in order to understand activity trends in metal-surface-catalyzed reactions in terms of the linear Brønsted–Evans–Polanyi relation and Hammer–Nørskov d-band model. In this paper, the d-band centers for eleven metals (Fe, Co, Ni, Cu, Ru, Rh, Pd, Ag, Ir, Pt, Au) and their pairwise bimetals for two different structures (1% metal doped- or overlayer-covered metal surfaces) are statistically predicted using machine learning methods from readily available values as descriptors for the target metals (such as the density and the enthalpy of fusion of each metal). The predictive accuracy of four regression methods with different numbers of descriptors and different test-set/training-set ratios are quantitatively evaluated using statistical cross validations. It is shown that the d-band centers are reasonably well predicted by the gradient boosting regression (GBR) method with only six descriptors, even when we predict 75% of the data from only 25% given for training (average root mean square error (RMSE) < 0.5 eV). This demonstrates a potential use of machine learning methods for predicting the activity trends of metal surfaces with a negligible CPU time compared to first-principles methods.

Received 18th February 2016

Accepted 23rd May 2016

DOI: 10.1039/c6ra04345c

[www.rsc.org/advances](http://www.rsc.org/advances)

# Let's reproduce a paper-level result!

Table 1 DFT calculated d-band centers (eV) of metals (italic) and 1% guest metals ( $M_g$ ) doped in the surface of host metals ( $M_h$ ) as reported by Nørskov's group<sup>1,2</sup>

$M_h$	$M_g$										
	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	<i>-0.92</i>	-0.87	-1.12	-1.05	-1.21	-1.46	-2.16	-1.75	-1.28	-2.01	-2.34
Co	-1.16	<i>-1.17</i>	-1.45	-1.33	-1.41	-1.75	-2.54	-2.08	-1.53	-2.36	-2.73
Ni	-1.20	-1.10	<i>-1.29</i>	-1.10	-1.43	-1.60	-2.26	-1.82	-1.43	-2.09	-2.42
Cu	-2.11	-2.07	-2.40	<i>-2.67</i>	-2.09	-2.35	-3.31	-3.37	-2.09	-3.00	-3.76
Ru	-1.20	-1.15	-1.40	-1.29	<i>-1.41</i>	-1.58	-2.23	-1.68	-1.39	-2.03	-2.25
Rh	-1.49	-1.39	-1.57	-1.29	-1.69	<i>-1.73</i>	-2.27	-1.66	-1.56	-2.08	-2.22
Pd	-1.46	-1.29	-1.33	-0.89	-1.59	-1.47	<i>-1.83</i>	-1.24	-1.30	-1.64	-1.66
Ag	-3.58	-3.46	-3.63	-3.83	-3.46	-3.44	-4.16	<i>-4.30</i>	-3.16	-3.80	-4.45
Ir	-1.90	-1.84	-2.06	-1.90	-2.02	-2.26	-2.84	-2.24	<i>-2.11</i>	-2.67	-2.85
Pt	-1.92	-1.77	-1.85	-1.53	-2.11	-2.02	-2.42	-1.81	-1.87	<i>-2.25</i>	-2.30
Au	-2.93	-2.79	-2.93	-3.01	-2.86	-2.81	-3.39	-3.35	-2.58	-3.10	<i>-3.56</i>

Table 3 Input features (descriptors) used for prediction of d-band centers from ref. 34<sup>a</sup>

Metal	G	$R/\text{\AA}$	AN	AM/g mol <sup>-1</sup>	P	EN	IE/eV	$\Delta_{\text{fus}}H/J \text{ g}^{-1}$	$\rho/\text{g cm}^{-3}$
Fe	8	2.66	26	55.85	4	1.83	7.90	247.3	7.87
Co	9	2.62	27	58.93	4	1.88	7.88	272.5	8.86
Ni	10	2.60	28	58.69	4	1.91	7.64	290.3	8.90
Cu	11	2.67	29	63.55	4	1.90	7.73	203.5	8.96
Ru	8	2.79	44	101.07	5	2.20	7.36	381.8	12.10
Rh	9	2.81	45	102.91	5	2.28	7.46	258.4	12.40
Pd	10	2.87	46	106.42	5	2.20	8.34	157.3	12.00
Ag	11	3.01	47	107.87	5	1.93	7.58	104.6	10.50
Ir	9	2.84	77	192.22	6	2.20	8.97	213.9	22.50
Pt	10	2.90	78	195.08	6	2.20	8.96	113.6	21.50
Au	11	3.00	79	196.97	6	2.40	9.23	64.6	19.30

Table 1 and Table 3



Fig 1

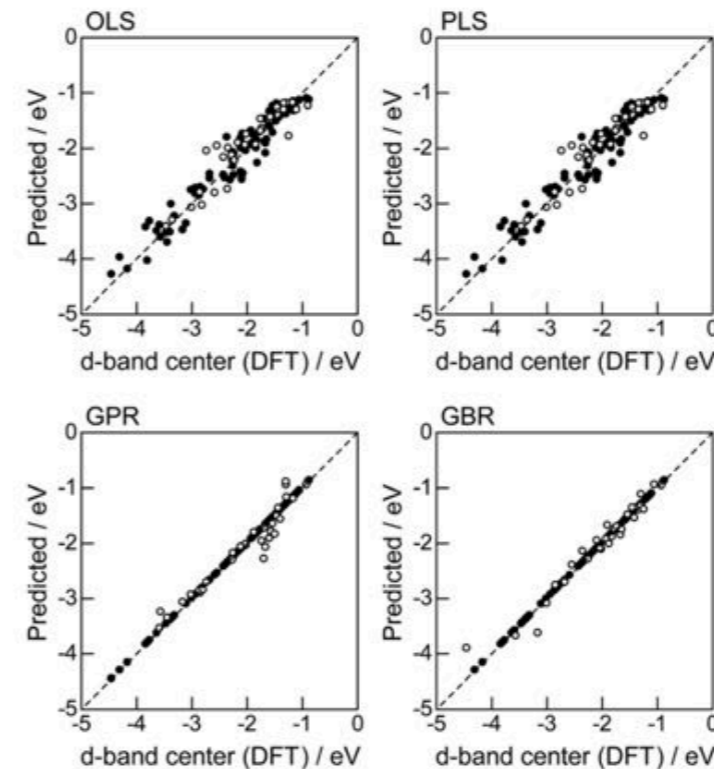
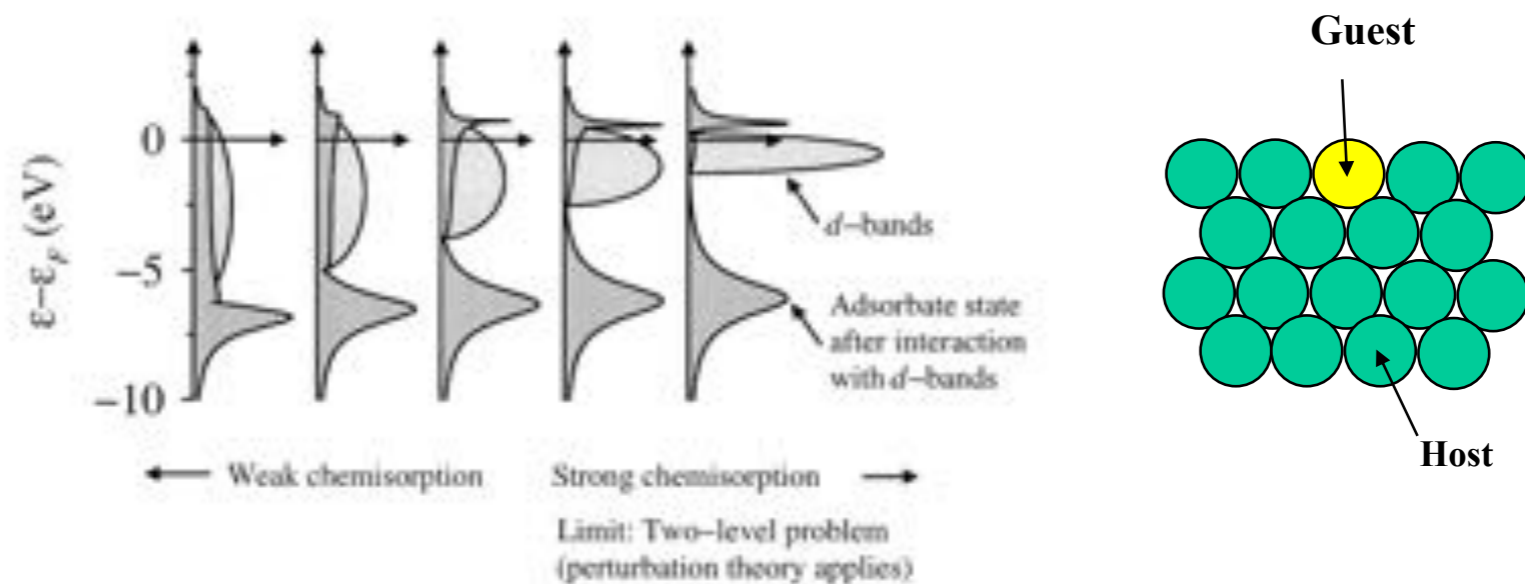


Fig. 1 DFT calculated local d-band center for metals and 1% guest metal-doped metals (Table 1) and the values predicted by linear (OLS, PLS) and nonlinear regression (GPR, GBR): (●) training set = 75%, (○) test set = 25%.

# Let's reproduce a paper-level result!

**Table 1: the energy of the d-band center relative to the Fermi level ( $\epsilon_F$ ),  $\epsilon - \epsilon_F$  for 1% guest metals doped in the surface of host metals.**



calculated from the original data

A. Ruban, B. Hammer, P. Stoltze, H. L. Skriver and J. K. Nørskov, J. Mol. Catal. A: Chem., 1997, 115, 421–429.

Table 1

Shifts in d-band centers of surface impurities and overlayers relative to the clean metal values (italic)

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09	-1.42
Co	0.01	-1.17	-0.28	-0.16	-0.24	-0.58	-1.37	-0.91	-0.36	-1.19	-1.56
Ni	0.09	0.19	-1.29	0.19	-0.14	-0.31	-0.97	-0.53	-0.14	-0.80	-1.13
Cu	0.56	0.60	0.27	-2.67	0.58	0.32	-0.64	-0.70	0.58	-0.33	-1.09
Ru	0.21	0.26	0.01	0.12	-1.41	-0.17	-0.82	-0.27	0.02	-0.62	-0.84
Rh	0.24	0.34	0.16	0.44	0.04	-1.73	-0.54	0.07	0.17	-0.35	-0.49
Pd	0.37	0.54	0.50	0.94	0.24	0.36	-1.83	0.59	0.53	0.19	0.17
Ag	0.72	0.84	0.67	0.47	0.84	0.86	0.14	-4.30	1.14	0.50	-0.15
Ir	0.21	0.27	0.05	0.21	0.09	-0.15	-0.73	-0.13	-2.11	-0.56	-0.74
Pt	0.33	0.48	0.40	0.72	0.14	0.23	-0.17	0.44	0.38	-2.25	-0.05
Au	0.63	0.77	0.63	0.55	0.70	0.75	0.17	0.21	0.98	0.46	-3.56
	0.53	0.74	0.71	0.70	0.47	0.67	0.35	0.12	0.79	0.43	

The impurity/overlayer atoms are listed horizontally and the host entries are listed vertically. For each combination of the two numbers listed is first the isolated surface impurity given and then the overlayer. The surfaces considered are the most close packed and the overlayer structures are pseudomorphic. No relaxations from the host lattice positions are included. All values are in eV and the elemental d band centers are relative to the Fermi level.

# Let's reproduce a paper-level result!

You can get the  part of the original data

```
[1] import numpy as np
import pandas as pd
```

```
[2] url = "https://itakigawa.github.io/data/hsi2020/data\_impurities.csv"
my_table = pd.read_csv(url, index_col=0)
```

```
[3] my_table
```

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09	-1.42
Co	0.01	-1.17	-0.28	-0.16	-0.24	-0.58	-1.37	-0.91	-0.36	-1.19	-1.56
Ni	0.09	0.19	-1.29	0.19	-0.14	-0.31	-0.97	-0.53	-0.14	-0.80	-1.13
Cu	0.56	0.60	0.27	-2.67	0.58	0.32	-0.64	-0.70	0.58	-0.33	-1.09
Ru	0.21	0.26	0.01	0.12	-1.41	-0.17	-0.82	-0.27	0.02	-0.62	-0.84



# handle table data by "pandas"

```
▶ my_table.loc['Co', 'Ni']
```

```
↳ -0.28
```

```
▶ my_table.loc['Co']
```

```
↳ Fe      0.01  
   Co     -1.17  
   Ni     -0.28  
   Cu     -0.16  
   Ru     -0.24  
   Rh     -0.58  
   Pd     -1.37  
   Ag     -0.91  
   Ir     -0.36  
   Pt     -1.19  
   Au     -1.56  
Name: Co, dtype: float64
```

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09	-1.42
Co	0.01	-1.17	-0.28	-0.16	-0.24	-0.58	-1.37	-0.91	-0.36	-1.19	-1.56
Ni	0.09	0.19	-1.29	0.19	-0.14	-0.31	-0.97	-0.53	-0.14	-0.80	-1.13
Cu	0.56	0.60	0.27	-2.67	0.58	0.32	-0.64	-0.70	0.58	-0.33	-1.09
Ru	0.21	0.26	0.01	0.12	-1.41	-0.17	-0.82	-0.27	0.02	-0.62	-0.84

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09	-1.42
Co	0.01	-1.17	-0.28	-0.16	-0.24	-0.58	-1.37	-0.91	-0.36	-1.19	-1.56
Ni	0.09	0.19	-1.29	0.19	-0.14	-0.31	-0.97	-0.53	-0.14	-0.80	-1.13
Cu	0.56	0.60	0.27	-2.67	0.58	0.32	-0.64	-0.70	0.58	-0.33	-1.09
Ru	0.21	0.26	0.01	0.12	-1.41	-0.17	-0.82	-0.27	0.02	-0.62	-0.84

# handle table data by "pandas"

```
▶ for h in my_table.index:  
    for g in my_table.columns:  
        print(f'host {h}, guest {g}, val {my_table.loc[h, g]}')
```

```
↳ host Fe, guest Fe, val -0.92  
host Fe, guest Co, val 0.05  
host Fe, guest Ni, val -0.2  
host Fe, guest Cu, val -0.13  
host Fe, guest Ru, val -0.29  
host Fe, guest Rh, val -0.54
```

columns

index

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09	-1.42
Co	0.01	-1.17	-0.28	-0.16	-0.24	-0.58	-1.37	-0.91	-0.36	-1.19	-1.56
Ni	0.09	0.19	-1.29	0.19	-0.14	-0.31	-0.97	-0.53	-0.14	-0.80	-1.13
Cu	0.56	0.60	0.27	-2.67	0.58	-0.32	-0.64	-0.70	0.58	-0.33	-1.09
Ru	0.21	0.26	0.01	0.12	-1.41	-0.17	-0.82	-0.27	0.02	-0.62	-0.84

# Get Table 1

A. Ruban, B. Hammer, P. Stoltze, H. L. Skriver and J. K. Nørskov, J. Mol. Catal. A: Chem., 1997, 115, 421–429.

Table 1

Shifts in d-band centers of surface impurities and overlayers **relative to the clean metal values (italic)**

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt
Fe	<b>-0.92</b>	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09

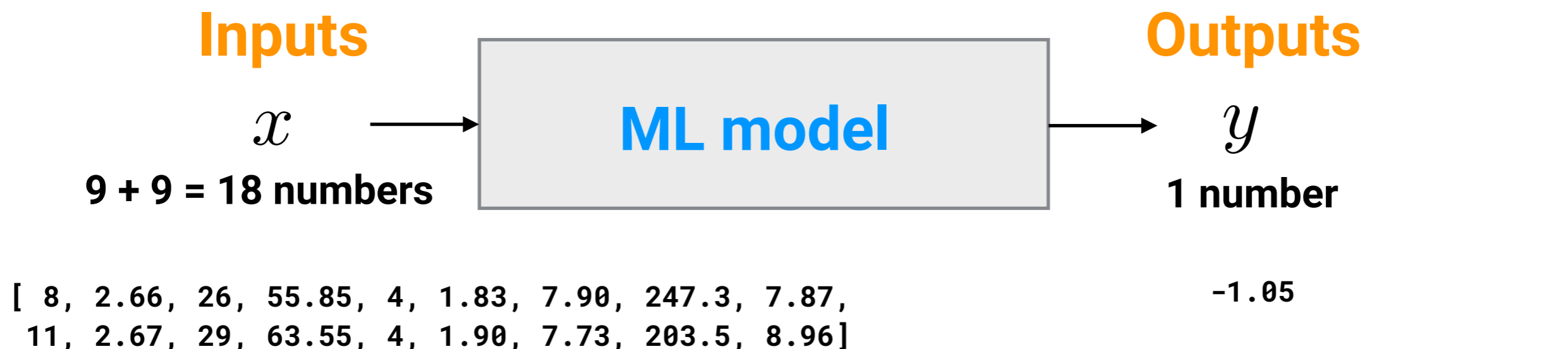
```
▶ for h in my_table.index:  
    for g in my_table.columns:  
        if h != g:  
            my_table.loc[h, g] += my_table.loc[h, h]
```

my\_table

```
↳
```

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	-0.87	-1.12	-1.05	-1.21	-1.46	-2.16	-1.75	-1.28	-2.01	-2.34
Co	-1.16	-1.17	-1.45	-1.33	-1.41	-1.75	-2.54	-2.08	-1.53	-2.36	-2.73
Ni	-1.20	-1.10	-1.29	-1.10	-1.43	-1.60	-2.26	-1.82	-1.43	-2.09	-2.42
Cu	-2.11	-2.07	-2.40	-2.67	-2.09	-2.35	-3.31	-3.37	-2.09	-3.00	-3.76
Ru	-1.20	-1.15	-1.40	-1.29	-1.41	-1.58	-2.23	-1.68	-1.39	-2.03	-2.25

# Our goal here is the following machine learning.



9 features (host metal) from Table 3  
+  
9 features (guest metal) from Table 3

the value at Table 1  
for the host and guest metal

Table 3 Input features (descriptors) used for prediction of d-band centers from ref. 34<sup>a</sup>

Metal	G	R/Å	AN	AM/g mol <sup>-1</sup>	P	EN	IE/eV	$\Delta_{\text{fus}}H/J$ g <sup>-1</sup>	$\rho/g$ cm <sup>-3</sup>
Fe	8	2.66	26	55.85	4	1.83	7.90	247.3	7.87
Co	9	2.62	27	58.93	4	1.88	7.88	272.5	8.86
Ni	10	2.60	28	58.69	4	1.91	7.64	290.3	8.90
Cu	11	2.67	29	63.55	4	1.90	7.73	203.5	8.96

	Fe	Co	Ni	Cu	Ru	Rh
Fe	-0.92	-0.87	-1.12	-1.05	-1.21	-1.46
Co	-1.16	-1.17	-1.45	-1.33	-1.41	-1.75
Ni	-1.20	-1.10	-1.29	-1.10	-1.43	-1.60
Cu	-2.11	-2.07	-2.40	-2.67	-2.09	-2.35

# Let's make the inputs

```
[17] url2 = "https://itakigawa.github.io/data/hai2020/features9.csv"  
      feat = pd.read_csv(url2, index_col=0)
```

```
[18] feat.head(3)
```



	name	Num of d- electrons	Bulk wigner- seitz radius	atomic number	atomic mass	period	electronegativity	Ionization energy (eV)
symbol								
Fe	Iron	8	2.66	26	55.8450	4	1.83	7.9024
Co	Cobalt	9	2.62	27	58.9332	4	1.88	7.8810
Ni	Nickel	10	2.60	28	58.6934	4	1.91	7.6398

```
[19] feat.loc['Co']
```



```
name          Cobalt  
Num of d-electrons      9  
Bulk wigner-seitz radius 2.62  
atomic number      27
```

# Let's make the inputs

```
feat.drop('name', axis='columns', inplace=True)
```

```
X = list()
y = list()
for h in my_table.index:
    for g in my_table.columns:
        vec_h = feat.loc[h].to_numpy()
        vec_g = feat.loc[g].to_numpy()
        x_val = np.concatenate((vec_h, vec_g))
        y_val = my_table.loc[h][g]
        X.append(x_val)
        y.append(y_val)
    if h == 'Fe' and g == 'Cu':
        print(f'host({h}), guest({g}), input={x_val}, output={y_val}')
```

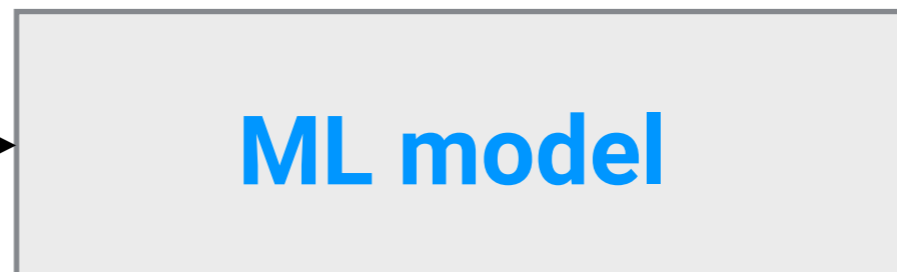
```
host(Fe), guest(Cu), input=[ 8.          2.66   26.          55.845    4.          1.83    7.9024 247.3
 7.87   11.          2.67   29.          63.546    4.          1.9    7.7264
203.5    8.96 ], output=-1.05
```



**Inputs**

$x$

9 + 9 = 18 numbers



**Outputs**

$y$

1 number

[ 8, 2.66, 26, 55.85, 4, 1.83, 7.90, 247.3, 7.87,  
11, 2.67, 29, 63.55, 4, 1.90, 7.73, 203.5, 8.96]

-1.05

# Our (X, y) data



```
df = pd.DataFrame(X, columns=[f'x{i+1}' for i in range(18)])  
df['y'] = y  
df
```

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	y
0	11.0	2.67	29.0	63.54600	4.0	1.90	7.7264	203.5	8.96	9.0	2.84	77.0	192.21700	6.0	2.20	8.9670	213.9	22.50	-2.09
1	9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	11.0	3.00	79.0	196.96655	6.0	2.40	9.2255	64.6	19.30	-2.73
2	10.0	2.90	78.0	195.07800	6.0	2.20	8.9588	113.6	21.50	8.0	2.79	44.0	101.07000	5.0	2.20	7.3605	381.8	12.10	-2.11
3	11.0	3.00	79.0	196.96655	6.0	2.40	9.2255	64.6	19.30	9.0	2.84	77.0	192.21700	6.0	2.20	8.9670	213.9	22.50	-2.58
4	10.0	2.90	78.0	195.07800	6.0	2.20	8.9588	113.6	21.50	9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	-1.77
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
116	9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	9.0	2.81	45.0	102.90550	5.0	2.28	7.4589	258.4	12.40	-1.75
117	9.0	2.81	45.0	102.90550	5.0	2.28	7.4589	258.4	12.40	11.0	3.00	79.0	196.96655	6.0	2.40	9.2255	64.6	19.30	-2.22
118	8.0	2.79	44.0	101.07000	5.0	2.20	7.3605	381.8	12.10	9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	-1.15
119	9.0	2.81	45.0	102.90550	5.0	2.28	7.4589	258.4	12.40	8.0	2.79	44.0	101.07000	5.0	2.20	7.3605	381.8	12.10	-1.69
120	9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	9.0	2.84	77.0	192.21700	6.0	2.20	8.9670	213.9	22.50	-1.53

121 rows x 19 columns

# The full code for preparation (only 21 lines!)

```
▶ import numpy as np
import pandas as pd

url1 = "https://itakigawa.github.io/data/hsi2020/data\_impurities.csv"
url2 = "https://itakigawa.github.io/data/hsi2020/features9.csv"

my_table = pd.read_csv(url1, index_col=0)

feat = pd.read_csv(url2, index_col=0)
feat.drop('name', axis='columns', inplace=True)

for h in my_table.index:
    for g in my_table.columns:
        if h != g:
            my_table.loc[h, g] += my_table.loc[h, h]

X = list()
y = list()
for h in my_table.index:
    for g in my_table.columns:
        vec_h = feat.loc[h].to_numpy()
        vec_g = feat.loc[g].to_numpy()
        X.append(np.concatenate((vec_h, vec_g)))
        y.append(my_table.loc[h][g])

X = np.stack(X)
y = np.array(y)
```



# Now we can move on to the "machine learning" part!

```
[8] X.shape, y.shape
```

```
↳ ((121, 18), (121,))
```

The number of input-output examples is 121. So we'll use random 30 examples for evaluation, and the remaining 91 examples for the model fitting.

```
[21] from sklearn.utils import shuffle
      X, y = shuffle(X, y)
      X_train, y_train = X[:-30, :], y[:-30]
      X_test, y_test = X[-30:, :], y[-30:]
```

This shuffling is quite important.

See what happens if you skip it, and think why.

Let's go with "machine learning" with 1 line of "model.fit"

```
[23] from sklearn.ensemble import GradientBoostingRegressor
      model = GradientBoostingRegressor()
      model.fit(X_train, y_train)
      y_pred_train = model.predict(X_train)
      y_pred_test = model.predict(X_test)
```

# Make a plot.

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.scatter(y_train, y_pred_train, \
          alpha=0.5, color="blue", label="training")
ax.scatter(y_test, y_pred_test, \
          alpha=0.5, color="red", label="test")
ax.legend()
ax.set_xlabel('groudtruth')
ax.set_ylabel('prediction')
ax.set_aspect("equal")
```

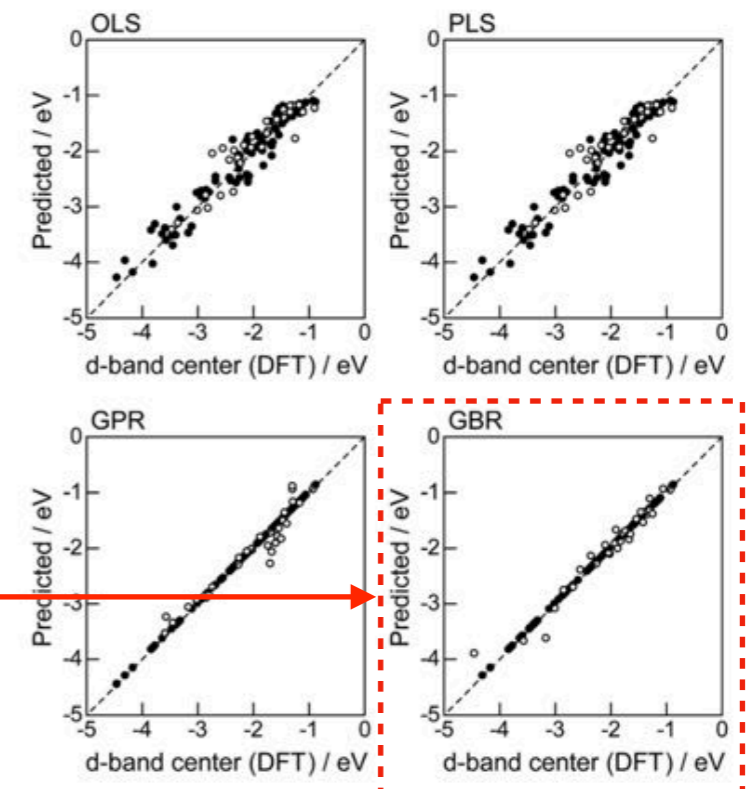
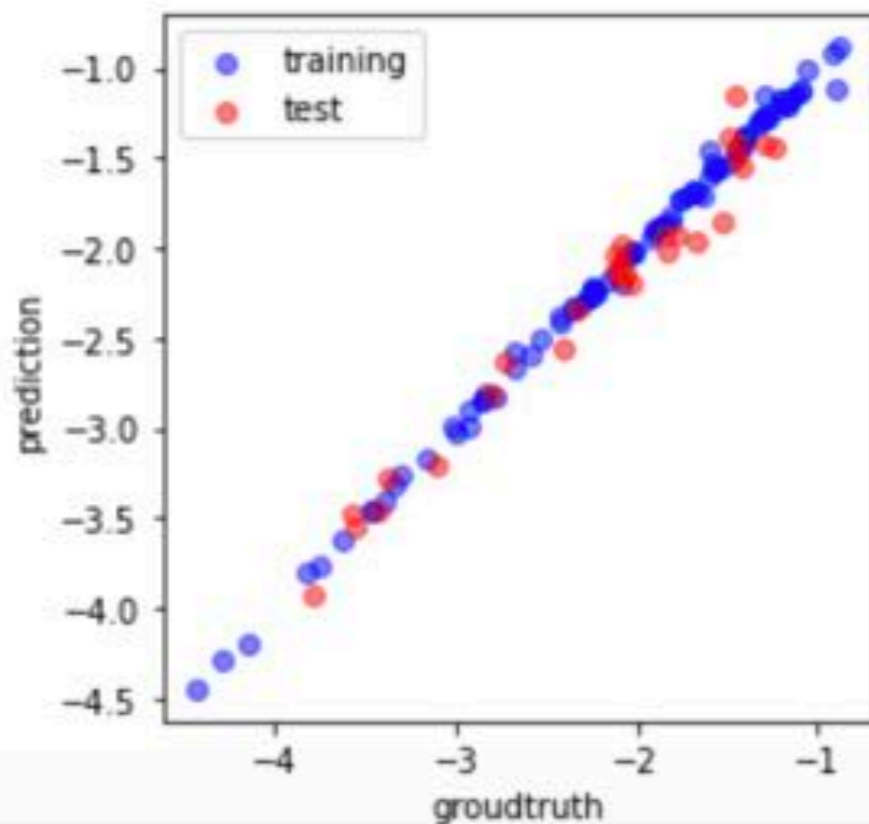


Fig. 1 DFT calculated local d-band center for metals and 1% guest metal-doped metals (Table 1) and the values predicted by linear (OLS, PLS) and nonlinear regression (GPR, GBR): (●) training set = 75%, (○) test set = 25%.

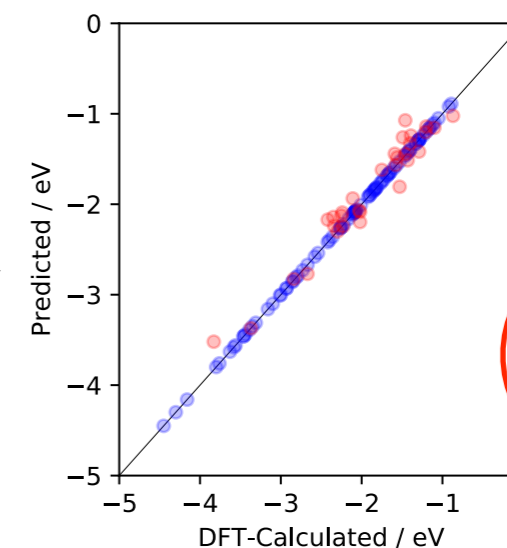
# The paper used only 6 features out of 18...

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92		-0.96	-0.97	-1.65	-1.64	-2.24		-1.87	-2.4	-3.11
Co			-1.37	-1.23		-2.12	-2.82	-2.53	-2.26		-3.56
Ni	-0.33	-1.18			-1.92	-2.03		-2.43	-2.15	-2.82	-3.39
Cu	-2.42		-2.49	-2.67	-2.89	-2.94				-3.82	-4.63
Ru	-1.11	-1.04	-1.12		-1.41		-1.88	-1.81	-1.54		-2.27
Rh	-1.42	-1.32		-1.51	-1.7	-1.73	-2.12	-1.81	-1.7	-2.18	-2.3
Pd	-1.47	-1.29	-1.29	-1.03		-1.58	-1.83	-1.68	-1.52	-1.79	
Ag	-3.75	-3.56	-3.62		-3.8		-4.03		-3.5	-3.93	-4.51
Ir	-1.78	-1.71	-1.78	-1.55		-2.14	-2.53	-2.2	-2.11	-2.6	-2.7
Pt			-1.71	-1.47	-2.13	-2.01	-2.23	-2.06	-1.96		-2.33
Au	-3.03	-2.82	-2.85			-2.89		-3.44			-3.56

gradient boosting  
w/ 6 descriptors



training sets (75%)  
test sets (25%)

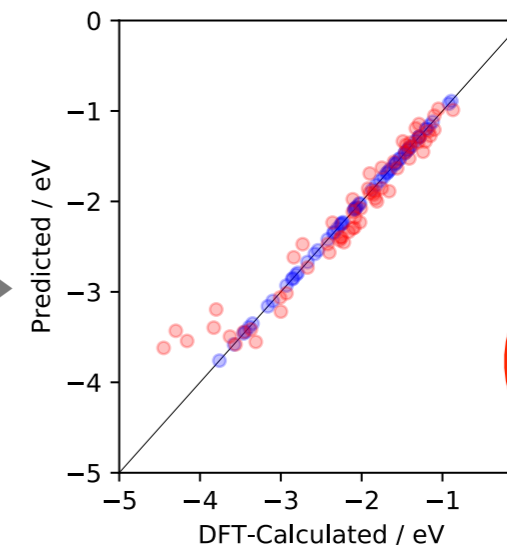


	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe		-0.78			-1.65	-1.64			-1.87		
Co	-1.18	-1.17	-1.37		-1.87	-2.12	-2.82		-2.26		
Ni	-0.33	-1.18		-1.17			-2.61	-2.43	-2.15	-2.82	
Cu	-2.42				-2.89	-2.94		-3.88			-4.63
Ru	-1.11	-1.04	-1.12	-1.11	-1.41			-1.81			-2.27
Rh	-1.42			-1.51			-2.12	-1.81	-1.7		
Pd		-1.29	-1.29	-1.03		-1.58	-1.83		-1.52	-1.79	
Ag				-3.68	-3.8	-3.63					-4.51
Ir						-2.14			-2.11		-2.7
Pt			-1.71	-1.47	-2.13	-2.01	-2.23	-2.06			
Au				-2.86	-3.09	-2.89		-3.44			-3.56

gradient boosting  
w/ 6 descriptors



training sets (50%)  
test sets (50%)

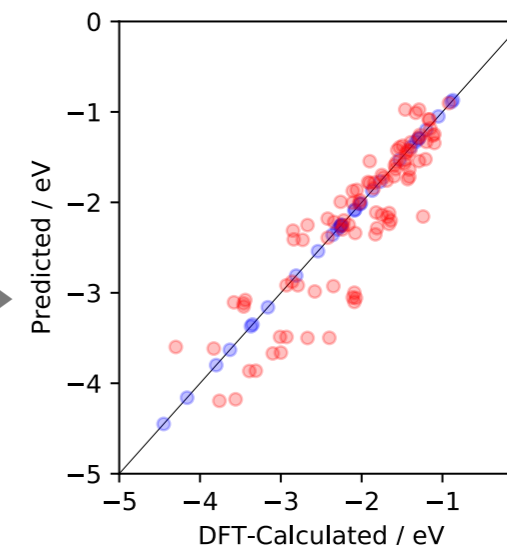


	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe								-2.17			-3.11
Co		-1.17	-1.37			-2.12					
Ni	-0.33	-1.18					-2.61	-2.43			
Cu	-2.42	-2.29	-2.49				-3.71				-4.63
Ru										-2.02	
Rh		-1.32				-1.73	-2.12				
Pd					-1.94		-1.83				-1.97
Ag	-3.75			-3.68							-4.51
Ir	-1.78	-1.71									-2.7
Pt					-2.13						
Au					-3.09	-2.89					

gradient boosting  
w/ 6 descriptors



training sets (25%)  
test sets (75%)



# We can also easily compute the quantitative performance!

[https://en.wikipedia.org/wiki/Root-mean-square\\_deviation](https://en.wikipedia.org/wiki/Root-mean-square_deviation)

```
▶ from sklearn.metrics import mean_squared_error
rmse_tr = mean_squared_error(y_train, y_pred_train, squared=False)
rmse_te = mean_squared_error(y_test, y_pred_test, squared=False)
print(f'RMSE(training) {rmse_tr:.3f}')
print(f'RMSE(test) {rmse_te:.3f}')
```

```
↳ RMSE(training) 0.041
   RMSE(test) 0.141
```

```
[34] from sklearn.model_selection import cross_val_score, ShuffleSplit
cvf = ShuffleSplit(n_splits=100, test_size=0.25)
scores = cross_val_score(model, X, y, cv=cvf, \
                          scoring='neg_root_mean_squared_error')
print(f'100 times mean RMSE: {-scores.mean():.3f}')
```

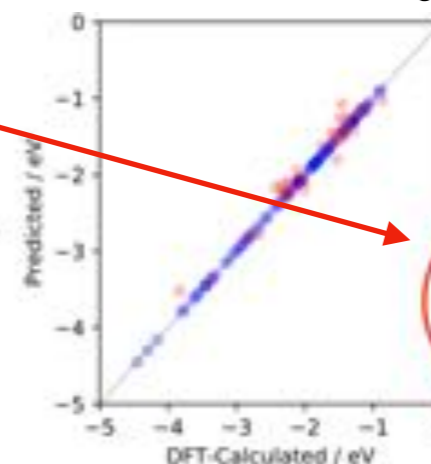
```
↳ 100 times mean RMSE: 0.153
```

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92		-0.96	-0.97	-1.65	-1.64	-2.24		-1.87	-2.4	-3.11
Co			-1.37	-1.23		-2.12	-2.82	-2.53	-2.26		-3.56
Ni	-0.33	-1.18			-1.92	-2.03		-2.43	-2.15	-2.82	-3.39
Cu	-2.42		-2.49	-2.67	-2.89	-2.94				-3.82	-4.63
Ru	-1.11	-1.04	-1.12		-1.41		-1.88	-1.81	-1.54		-2.27
Rh	-1.42	-1.32		-1.51	-1.7	-1.73	-2.12	-1.81	-1.7	-2.18	-2.3
Pd	-1.47	-1.29	-1.29	-1.03		-1.58	-1.83	-1.68	-1.52	-1.79	
Ag	-3.75	-3.56	-3.62		-3.8		-4.03		-3.5	-3.93	-4.51
Ir	-1.78	-1.71	-1.78	-1.55		-2.14	-2.53	-2.2	-2.11	-2.6	-2.7
Pt			-1.71	-1.47	-2.13	-2.01	-2.23	-2.06	-1.96		-2.33
Au	-3.03	-2.82	-2.85			-2.89		-3.44			-3.56

gradient boosting  
w/ 6 descriptors



training sets (75%)  
test sets (25%)



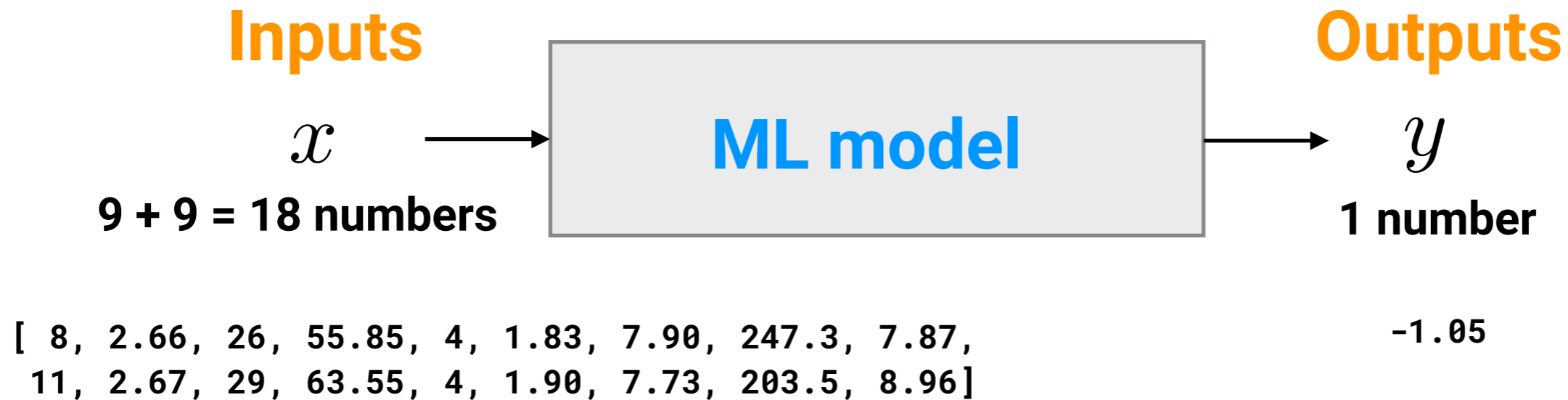
Do this calculation over  
100 random splits into 75%  
/ 25% subsets, and take the  
average of 100 RMSEs.

root-mean-square error  
(RMSE) =  $\sqrt{\text{RMSE}}$  =

$$\sqrt{\frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}}$$

100 times  
mean RMSE:  
**0.153 / eV**


# Ignoring all problem-specific data preparation...



```
[23] from sklearn.ensemble import GradientBoostingRegressor  
model = GradientBoostingRegressor()  
model.fit(X_train, y_train)
```

↑  
└─ "machine learning" part is only here!

# <https://scikit-learn.org/stable/>

 [Install](#) [User Guide](#) [API](#) [Examples](#) [More](#) Go

## scikit-learn

Machine Learning in Python

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

[Getting Started](#) [Release Highlights for 0.23](#) [GitHub](#)

### Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, and more...



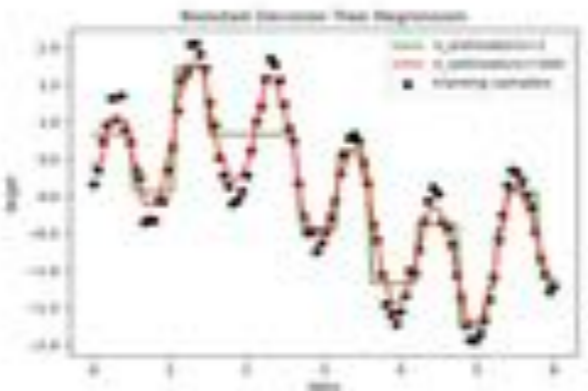
Examples

### Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.

**Algorithms:** SVR, nearest neighbors, random forest, and more...



Examples

### Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, spectral clustering, mean-shift, and more...



Examples

### Dimensionality reduction

Reducing the number of random variables to consider.

**Applications:** Visualization, increased efficiency

**Algorithms:** k-Means, feature selection, non-negative matrix factorization, and more...

### Model selection

Comparing, validating and choosing parameters and models.

**Applications:** Improved accuracy via parameter tuning

**Algorithms:** grid search, cross validation,

### Preprocessing

Feature extraction and normalization.

**Applications:** Transforming input data such as text for use with machine learning algorithms.

**Algorithms:** preprocessing, feature extraction, and more...

## Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

## Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
  - "The quality of your inputs decide the quality of your output"
  - Training / validation / test data
  - Tuning hyperparameters
  - Identification and design of input variables (or "descriptors")
  - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions